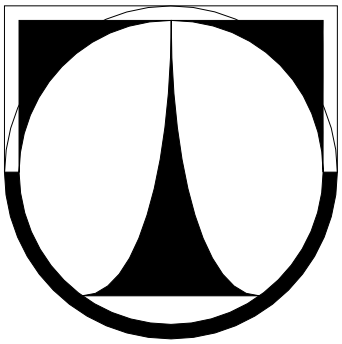


TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Řízení tepelné soustavy pomocí PLC Siemens

The Control of Heating Model by PLC Siemens

Bakalářská práce

Autor: **Martin Kopal**
Vedoucí práce: Ing. Lukáš Hubka, Ph.D.
Konzultant: Ing. Petr Školník, Ph.D.

V Liberci 14. 5. 2012

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Kopal**
Osobní číslo:	**M09000066**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Řízení tepelné soustavy pomocí PLC Siemens**
Zadávající katedra:	**Ústav řízení systémů a spolehlivosti**

Zásady pro vypracování:

1. Zjistěte statické i dynamické vlastnosti laboratorního systému. Zvolte vhodný pracovní bod/oblast.
2. Navrhněte decentralizovaný způsob řízení pomocí PID regulátorů a implementujte jej pomocí PLC Siemens.
3. Pokuste se do PLC aplikovat i některou z dalších metod řízení vícerozměrových systémů (např. některou z metod dekompozice).
4. Pokuste se zjistit, jaké jsou možnosti online monitoringu či exportu dat do PC při stávajícím HW vybavení a pokuste se o realizaci takového přenosu dat.

Rozsah grafických prací:	**dle potřeby**
Rozsah pracovní zprávy:	**cca 40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

1. **BERGER, H. Automatizace se STEPem 7 v AWL. München: Publicis MCD Verlag, 1998. 327 s.**
2. **HLAVA, J. Prostředky automatického řízení II. Praha: ČVUT, 2000. 162 s. Dostupné z WWW: <http://www.fm.tul.cz/~jaroslav.hlava/par/Skripta_PAR.pdf>.**
3. **MELICHAR, Jiří. Decentralizované a hierarchické řízení. Plzeň: KKY, 2010. 122 s.**
4. **HERGENHAHN, T. LIBNODAVE, a free communication library for Simatic S7 PLCs [online]. 2011 [cit. 2011-10-10]. Dostupné z WWW: <http://libnodave.sourceforge.net>.**

Vedoucí bakalářské práce:	**Ing. Lukáš Hubka, Ph.D.** Ústav řízení systémů a spolehlivosti
Konzultant bakalářské práce:	**Ing. Petr Školník, Ph.D.** Ústav řízení systémů a spolehlivosti
Datum zadání bakalářské práce:	**14. října 2011**
Termín odevzdání bakalářské práce:	**18. května 2012**

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Děkuji především vedoucímu bakalářské práce Ing. Lukáši Hubkovi, Ph.D. za odbornou pomoc a věcné rady s řešením různých problémů při zpracování bakalářské práce.

Abstrakt

Bakalářská práce se zabývá využitím programovatelného logického automatu od firmy Siemens pro řízení vícerozměrového systému, kterým je tepelná soustava tvořená dvěma vstupy a dvěma výstupy. V bakalářské práci je čtenář seznámen s návrhem vhodných regulačních struktur ve formě decentralizovaného řízení nebo případné dekompozice pro optimální regulaci daného vícerozměrového systému. Práce dále popisuje možnosti implementace vytvořených regulátorů a postupů řízení do hlavní řídící jednotky PLC. Nakonec se zabývá komunikací mezi počítačem a PLC s využitím OPC serveru a programu Reliance, kde pomocí těchto programů je umožněno sledování různých veličin v systému za provozu a případný záznam sledovaných dat.

Klíčová slova: PLC, regulace, vícerozměrový systém, monitoring

Abstract

The major point of this thesis is an application of programmable logic automat designed by Siemens company. This automat was used for control of multidimensional system, which is the thermal system formed by two inputs and two outputs. In the thesis the reader is introduced to the design of the appropriate regulative structures in the form of the decentralized control or the decomposition for the optimal regulation of the multidimensional system. In the next part of the thesis are described possibilities of application of the created regulators and processes of the control into the central processor unit of the PLC. The last part is focused on communication between the computer and the PLC. The communication is realized by the OPC server and program which is called Reliance. These programs allow to monitor different variables and recording monitored data.

Key words: PLC, regulation, multidimensional system, monitoring

Obsah

Seznam zkratek

PLC	Programovatelný Logický Kontrolér (Programmable Logic Controller)
Ti	integrační časová konstanta
Td	derivační časová konstanta
K	zesílení regulátoru
USB	rozšiřující port počítače (Universal Serial Bus)
MPI	komunikační rozhraní PLC Siemens (Multi-Point Interface)
CPU	Centrální Procesorová Jednotka (Central Processing Unit)
$w(t)$	žádaná veličina
$y(t)$	regulovaná veličina
$u(t)$	akční veličina
$e(t)$	regulační odchylka, rozdíl žádané a regulované veličiny
$ys(t)$	výstupní veličina ze soustavy
$d(t)$	poruchová veličina
$G(s)$	přenos soustavy
$R(s)$	přenos regulátoru
MIMO	soustava s více vstupy/výstupy (Multiple Input Multiple Output)
MISO	soustava s více vstupy a jedním výstupem (Multiple Input Single Output)
SISO	soustava s jedním vstupem/výstupem (Single Input Single Output)
$\boldsymbol{u}(t)$	vektor akčních zásahů
$\boldsymbol{d_m}(t)$	vektor měřených poruchových veličin
$\boldsymbol{d}(t)$	vektor neměřených poruchových veličin
$\boldsymbol{y}(t)$	vektor regulovaných veličin
$\boldsymbol{e}(t)$	vektor regulačních odchylek
$\boldsymbol{y_S}(t)$	vektor výstupní veličiny ze soustavy
$\boldsymbol{v_m}(t)$	vektor šumu měření
$\boldsymbol{Y}(s)$	vektor obrazů regulovaných veličin
$\boldsymbol{G}(s)$	přenosová matice mezi $\boldsymbol{Y}(s)$ a $\boldsymbol{U}(s)$
$\boldsymbol{R}(s)$	matice regulátorů
$\boldsymbol{U}(s)$	vektor obrazů akčních zásahů
$\boldsymbol{D_m}(s)$	vektor obrazů měřených poruchových veličin
$\boldsymbol{D}(s)$	vektor obrazů neměřených poruchových veličin
$\boldsymbol{F_{YU}}$	dílčí přenosová matice mezi $\boldsymbol{Y}(s)$ a $\boldsymbol{U}(s)$

$\boldsymbol{F_{YDM}}$	dílčí přenosová matice mezi $\boldsymbol{Y}(s)$ a $\boldsymbol{D_m}(s)$
$\boldsymbol{F_{YD}}$	dílčí přenosová matice mezi $\boldsymbol{Y}(s)$ a $\boldsymbol{D}(s)$
STL	instrukční list (Statement List)
LAD	struktura reléové logiky (Ladder Logic)
FBD	diagram funkčních bloků (Function Block Diagram)
SCL	strukturovaný text (Structured Control Language)
OPC	komunikační protokol OPC serveru

Seznam ilustrací

Úvod

Bakalářská práce se věnuje popisu a možnosti řízení vícerozměrového systému. V praxi se většinou setkáváme s regulovanými soustavami, které jsou vícerozměrové, respektive mají více vstupů a výstupů. Popis vícerozměrového systému se provádí pomocí identifikace statických a dynamických vlastností určeného systému. Proces identifikace je nejdůležitější fází v průběhu celého řízení, protože bez známého popisu soustavy by pozdější návrhy regulátorů a regulačních struktur byly zcela nemožné nebo alespoň velmi obtížné.

Pro řízení vícerozměrových systémů se nejčastěji využívají PID regulátory. V některých případech může být postačujícím řešením použití PI regulátoru. Pro návrh regulátorů se využívají spolehlivé postupy, které zahrnují několik návrhových metod.

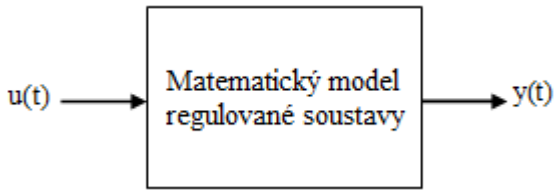
Úvodní část je věnována seznámení s vícerozměrovými systémy, jejich strukturou a obecným popisem. Dále se v této části nachází podrobnější popis soustavy, která je hlavním cílem řízení.

Ve druhé části se přiblíží různé návrhové metody regulátorů, mezi které patří např. metoda souhrnné časové konstanty, také označována jako Kuhnova metoda, případně experimentální nastavení regulátorů a další. Důležitými body v této části jsou možnosti řízení vícerozměrového systému, jakými jsou decentralizované řízení nebo dekompozice.

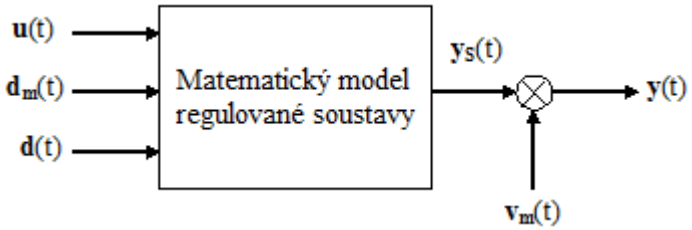
Poslední část se zabývá možností sledování různých parametrů v soustavě v reálném čase a jejich následným záznamem v některém z textových formátů.

1 Popis vícerozměrových systémů

Soustavy lze obecně rozdělit na dva základní typy. Jsou to soustavy SISO (obr. 1.1) a soustavy MIMO (obr. 1.2). Jednoznačně jednodušší pro regulaci jsou soustavy SISO, které disponují pouze jedinou vstupní a jedinou výstupní veličinou. Neexistují zde žádné dynamické křížové vazby, které by ovlivňovaly regulační pochod. Naopak je tomu u vícerozměrných systémů, kde přítomnost několika vstupních a výstupních veličin má za následek ovlivňování jednotlivých systémů mezi sebou. V ideálním případě by vstup jednoho systému působil pouze na výstup příslušejícího systému, ale v reálném případě je tomu zcela jinak, jeden vstup vícerozměrového systému, může působit i na několik výstupů ostatních systémů v soustavě. Z toho vyplývá, že mezi vstupními a výstupními veličinami existují různé dynamické křížové vazby, které je užitečné při řízení potlačit nebo alespoň částečně eliminovat.



Obr. 1.1 – Obecná struktura jednorozměrového systému



Obr. 1.2 – Obecná struktura vícerozměrového systému

Tyto dva základní typy systémů jsou v praxi nejvíce rozšířené, ale existuje ještě další varianta, která není tak častá, ale kombinuje vlastnosti obou systémů. Mezi tuto kombinaci patří systémy MISO, které mají využití např. v radiotechnice.

V klasických regulačních strukturách pro SISO systémy platí, že všechny veličiny, které do obvodu vstupují, vystupují nebo jistým způsobem zasahují (akční zásah, poruchová veličina, žádaná hodnota, regulovaná veličina a regulační odchylka) se popisují pomocí skaláru, a regulační obvod je tak pouze jednorozměrový. U vícerozměrových systémů ale platí, že všechny tyto veličiny se popisují pomocí vektorů.

Samotný vícerozměrový systém je popsán několika přenosovými funkcemi, jejichž počet je závislý na počtu regulovaných veličin a akčních zásahů, i na počtu křížových vazeb mezi regulovanými veličinami a akčními zásahy. Vícerozměrová soustava je tak tvořena maticí přenosů, která při stejném počtu vstupů a výstupů má tvar čtverce. Není pravidlem, že akční zásah jednoho systému má vliv na všechny regulované veličiny, to je dáno charakterem regulované soustavy, a proto některé křížové vazby mohou být nulové. Čím větší je takový počet křížových vazeb, které se neuplatňují, respektive jsou nulové, tím je v celkovém důsledku regulační struktura méně složitá.

Vícerozměrový systém lze popsat pomocí následující rovnice (1.1), která vychází z obecné struktury vícerozměrového systému, kde

$$\boldsymbol{Y}(s) = \left[\boldsymbol{F_{YU}}(s);\ \boldsymbol{F_{YD_m}}(s);\ \boldsymbol{F_{YD}}(s)\right] \cdot \begin{bmatrix} \boldsymbol{U}(s) \\ \boldsymbol{D_m}(s) \\ \boldsymbol{D}(s) \end{bmatrix} + \boldsymbol{V_m}(s) \tag{1.1}$$

Pokud se předpokládá, že vektory $\mathbf{D_m(s)}$, $\mathbf{D(s)}$ a $\mathbf{V_m(s)}$ jsou nulové, pak lze rovnici upravit do jednoduššího tvaru (1.2).

$$\boldsymbol{Y}(s) = \boldsymbol{F_{YU}}(s) \cdot \boldsymbol{U}(s) = \boldsymbol{G}(s) \cdot \boldsymbol{U}(s) \tag{1.2}$$

Z vyjádřené rovnice lze vidět závislost mezi vektorem vstupů a vektorem výstupů, kterou udává přenosová funkce. U vícerozměrového systému je přenosová funkce popsána maticí, jejíž jednotlivé složky lze obecně vyjádřit tímto způsobem.

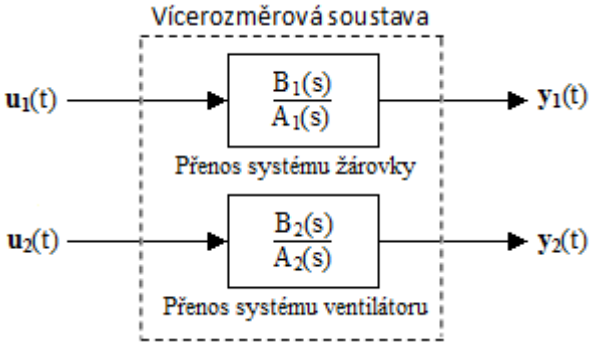
$$G_{ij}(s) = \frac{Y_i(s)}{U_j(s)} \tag{1.3}$$

Matice přenosů vyjadřuje závislost mezi i-tou výstupní veličinou a j-tou vstupní veličinou vícerozměrové soustavy. Obecný tvar matice přenosů lze ve formě matice reprezentovat následovně.

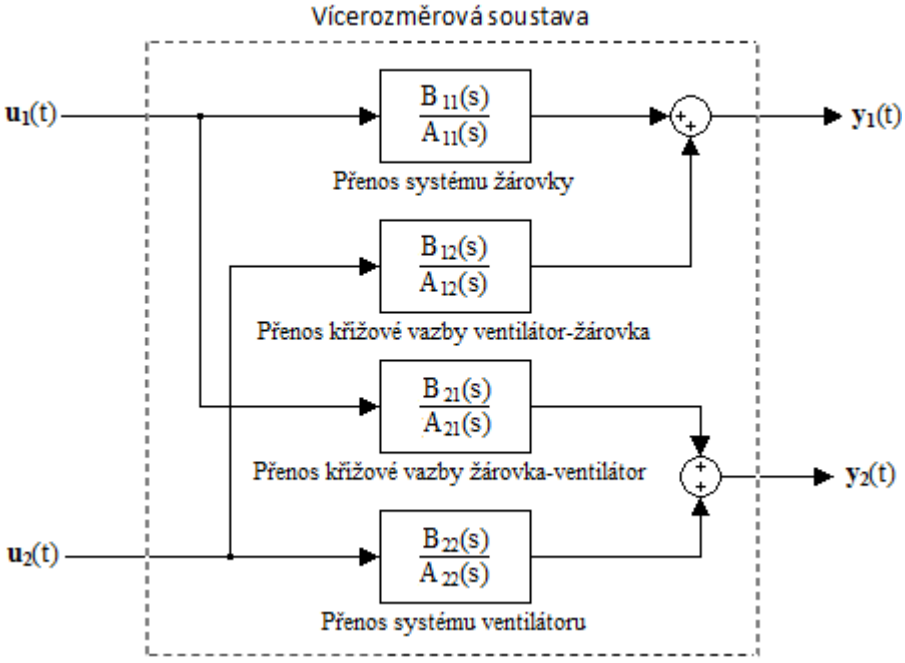
$$G_{ij} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1j} \\ G_{21} & G_{22} & \cdots & G_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ G_{i1} & G_{i2} & \cdots & G_{ij} \end{bmatrix}$$

1.1 Popis soustavy

Reálná vícerozměrová soustava (obr. 1.3), kterou je tepelná soustava a je cílem řízení, disponuje dvěma vstupy a dvěma výstupy, kterými jsou systém žárovky a ventilátoru. Jedna z možností, jak smýšlet o řízení tepelné soustavy, je uvažovat soustavu opravdu jako vícerozměrovou s daným počtem vstupů a výstupů. Zároveň je důležité uvědomit si, že vícerozměrová soustava může obsahovat různé křížové vazby mezi systémy (obr. 1.4), které ovlivňují celkovou dynamiku soustavy. U této reálné soustavy je významná křížová vazba mezi systémem ventilátoru a žárovky, kde proud vzduchu od ventilátoru velmi výrazně ovlivňuje teplotu na žárovce. Naopak druhá křížová vazba mezi žárovkou a ventilátorem se vůbec neuplatňuje a je proto nulová.

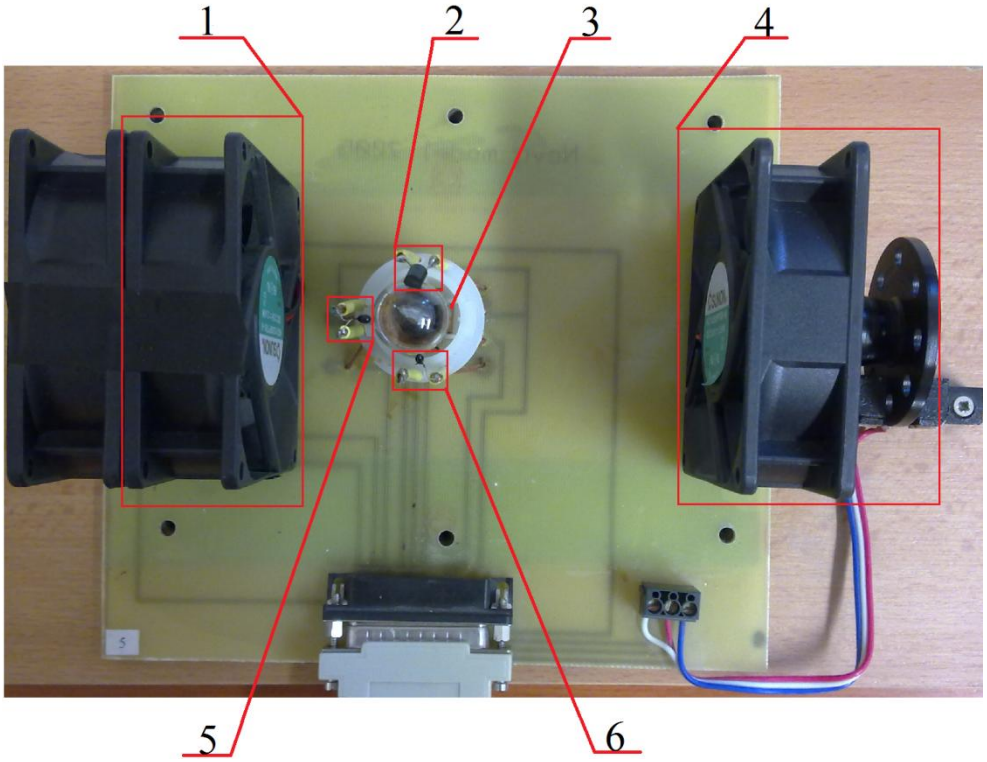


Obr. 1.3 - Obecná struktura reálné vícerozměrové soustavy



Obr. 1.4 – Obecná struktura reálné vícerozměrové soustavy s křížovými vazbami

Reálná vícerozměrová soustava (obr. 1.5) je složena z několika hlavních částí, které jsou důležité pro řízení a jsou podrobně rozebrány v následujících odstavcích. Pro detailnější pohled na reálnou soustavu jsem odstranil kryt, který tvoří aerodynamický tunel proudu vzduchu. První systém se skládá z ventilátoru (1), který je vstupem systému a slouží k vytváření požadovaného průtoku vzduchu. Ventilátor se řídí vstupním napětím v rozsahu 0–10 V. Výstupem prvního systému je druhý ventilátor (4), který slouží jako senzor průtoku vzduchu. Otáčky druhého ventilátoru jsou spojeny s plastovým kotoučkem, který není plný, ale má místy průchozí otvory, z důvodu snímání otáček pomocí inkrementálního čidla. Inkrementální čidlo je tvořeno infračervenou diodou snímající počet pulzů na otáčku, které jsou pak v převodníku převedeny na výstupní napětí v rozsahu 0–10 V.

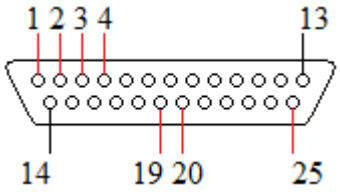


Obr. 1.5 – Reálná vícerozměrová soustava

Druhý systém se skládá z žárovky (3), která je vstupem systému a slouží k řízení teploty v soustavě. Žárovka je řízena vstupním napěťovým signálem v rozsahu 0–10 V. Výstup tohoto systému tvoří několik termistorů, které měří teplotu soustavy. První termistor (2) má typové označení *KTY 81 210* a je opatřen plastovým pouzdrem. Termistor je umístěn v minimální vzdálenosti od baňky žárovky. Další dva termistory (5) a (6) mají typové označení *NR 354 20K U* a tyto dva senzory mají pouze

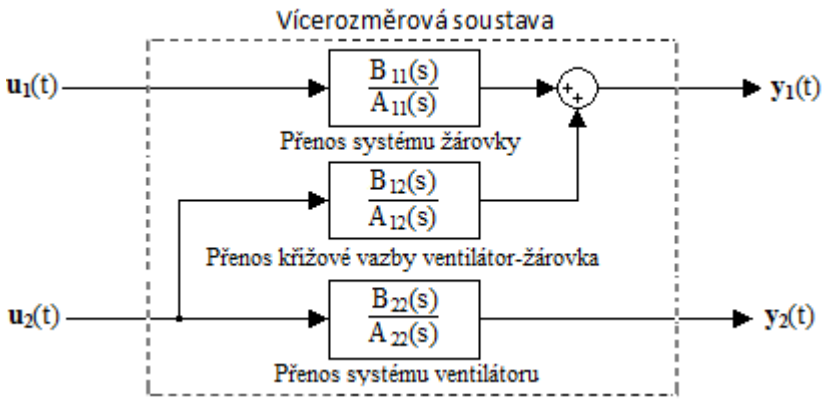
ochranné pouzdro tvořené pryskyřicí. Jeden termistor (6) je umístěn v minimální vzdálenosti od baňky žárovky a druhý termistor (5) je umístěn ve vzdálenosti 5 mm od baňky. Signály ze všech čidel teploty jsou v převodníku převedeny na výstupní napěťové hodnoty v rozsahu 0–10 V.

Při komunikaci mezi vícerozměrovou soustavou, převodníkem a mezi PLC kontrolérem se využívá sériových portů, které mají 25 pinů (obr. 1.6). Pro řízení vícerozměrové soustavy jsou nejdůležitější piny 1 až 4, 19, 20 a 25, kde piny 1 až 3 jsou spojeny se senzory teploty žárovky, pin 4 je spojen se senzorem průtoku vzduchu. Další dva piny 19 a 20 jsou využívány pro napájení vstupů jednotlivých systémů, respektive napájení ventilátoru a žárovky. Poslední pin 25 slouží jako zemnící vodič.



Obr. 1.6 – Sériový port

V důsledku toho, že jedna křížová vazba v reálné soustavě mezi žárovkou a ventilátorem se nemusí uvažovat (obr. 1.7), protože systém žárovky nemůže svojí teplotou ovlivňovat velikost průtoku vzduchu od ventilátoru, je tak možné pohlížet na reálnou soustavu i jako na jednorozměrový systém. Hlavním řízeným systémem by zůstal systém žárovky, respektive řízení teploty, ale silná křížová vazba mezi ventilátorem a žárovkou způsobuje výrazné ovlivňování teploty na žárovce. Proto by se ventilátor mohl uvažovat ve vícerozměrové reálné soustavě jako měřená porucha, kde na základě známé poruchy by se mohla navrhnout regulační struktura, která by rušivé vlivy způsobené ventilátorem potlačila.



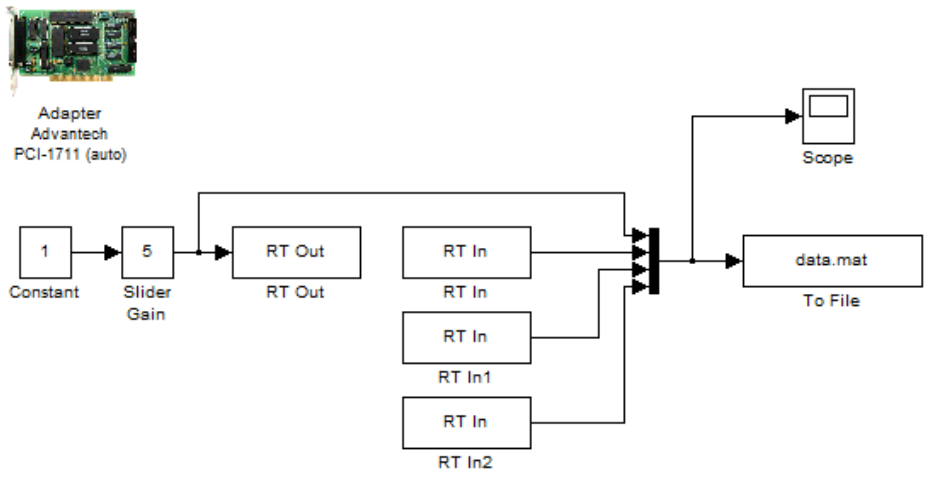
Obr. 1.7 – Obecná struktura reálné soustavy s nulovou křížovou vazbou

1.2 Identifikace soustavy

První seznámení se soustavou by mělo začínat bezpochyby její identifikací. Proces identifikace soustavy je nejdůležitější částí pro úspěšné řízení jednorozměrové nebo vícerozměrové soustavy. Jedná se o postup, který umožňuje zjistit chování daného systému, respektive jeho statické a dynamické vlastnosti. Výsledkem procesu identifikace je, že pomocí získaných dat lze následně určit matematické vyjádření odezvy systému ve formě přenosové funkce. Díky získané přenosové funkci se mohou provádět různé simulace systému bez toho, aby se muselo pracovat s reálným systémem, protože v principu přenosová funkce napodobuje reálný systém. Je možné tak navrhovat vhodné regulátory a optimální regulační struktury v simulačních podmínkách bez rizika poškození reálné soustavy. Následně je možné navržené regulátory a regulační struktury aplikovat na reálný systém a zjistit, zdali simulované odezvy odpovídají odezvám reálného systému, kde při správně provedené identifikaci by nemělo docházet k výrazným odchylkám. Samozřejmostí jsou odchylky v určité toleranci, protože nikdy není úplná jistota ideální identifikace. Systém se může vlivem okolních podmínek nebo vlivem změn různých vnitřních komponent v systému měnit a provedená identifikace tak po určité době nemusí zcela odpovídat. V případě, že by identifikace nebyla provedena, nebylo by možné provádět všechny popsané postupy a celkové řízení dané soustavy by bylo zcela nemožné nebo alespoň velmi obtížné.

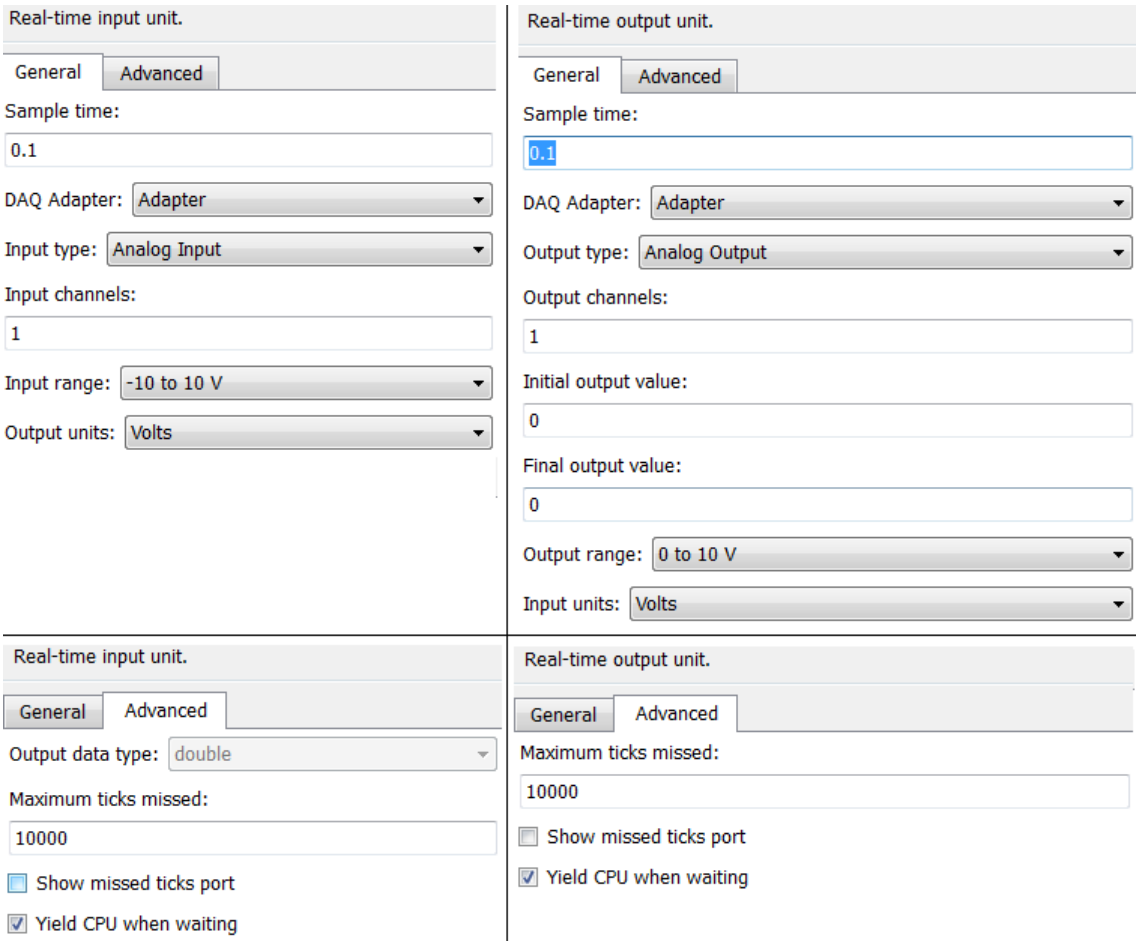
1.2.1 Schéma měření

Pro proces identifikace a porovnání různých reakcí reálného systému s reakcemi simulovanými jsem využíval speciální multifunkční měřicí kartu, která má typové označení *Adventech PCI-1711* a pro komunikaci s kartou jsem využíval programu Matlab, konkrétně podprogram Simulink, ve kterém jsem realizoval měřicí schéma (obr. 1.8). Nejdůležitější blok ve schématu je blok *Adapter*, který slouží k inicializaci příslušné multifunkční karty a pro komunikaci s kartou se využívají bloky *RT Out* a *RT In*.



Obr. 1.8 – Schéma měření pro systém žárovky

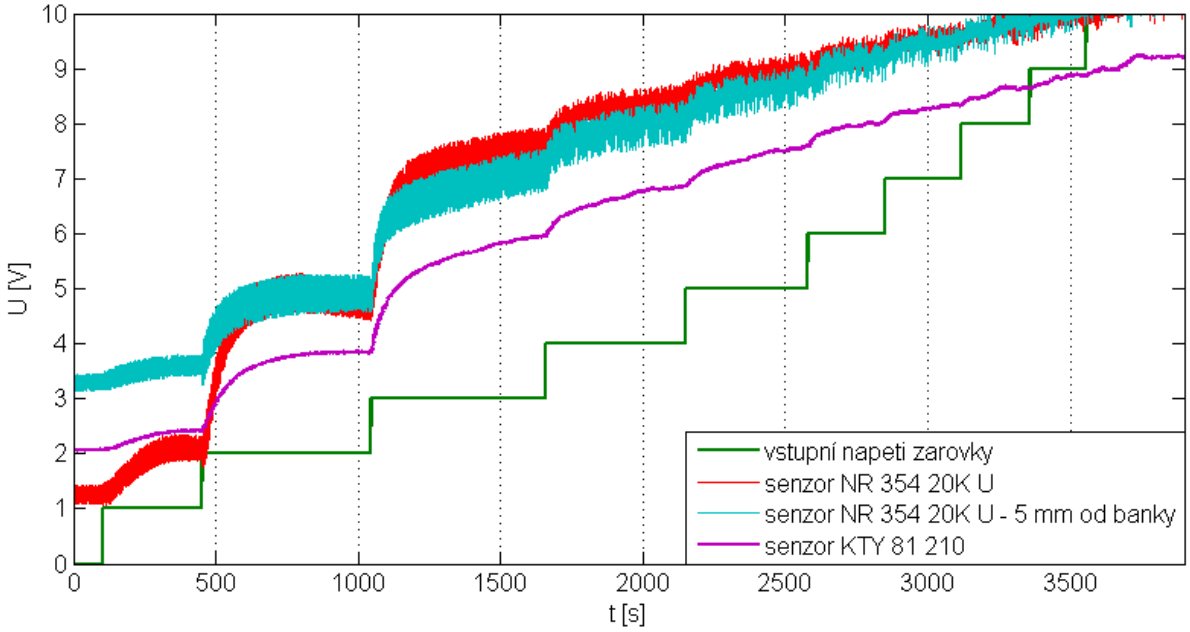
Bloky RT Out a RT In slouží k podrobnějšímu nastavení vstupních a výstupních kanálů karty (obr. 1.9). Umožňují nastavit typ vstupu a výstupu, jestli se budou využívat analogové nebo digitální kanály, v jakém rozsahu se bude měřit napětí a také důležitý parametr je doba vzorkování. Dobu vzorkování jsem volil většinou 0,1 s, případně při složitějších měřeních jsem volil dobu vzorkování 0,25 s. Je důležité nastavit dobu vzorkování stejnou u všech bloků, které se ve schématu měření používají pro čtení nebo zápis hodnot (bloky RT Out, RT In a To File). Blok RT Out ještě umožňuje nastavit hodnoty, které budou na výstupním kanálu v době spuštění měření a při ukončení.



Obr. 1.9 – Nastavení bloků *RT Out* a *RT In*

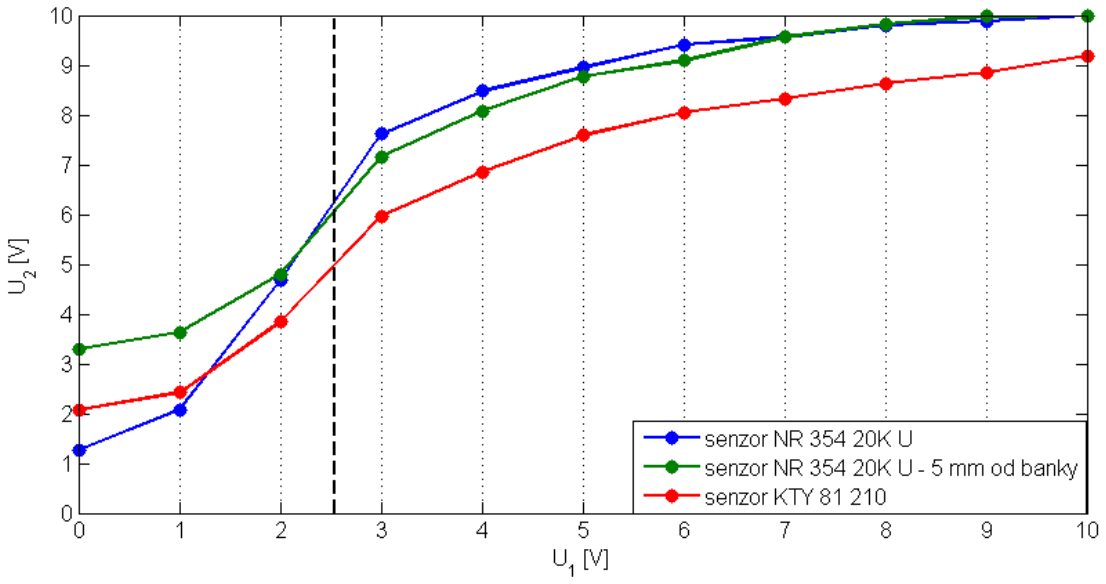
1.2.2 Statické vlastnosti

Statické vlastnosti popisují ustálený stav systému, který nastane po ukončení přechodového děje. Statika systému je tak popsána pomocí statické charakteristiky, která udává závislost ustálené hodnoty výstupu na ustálené hodnotě vstupu. Příslušnou definicí jsem se řídil při měření statických charakteristik obou systémů v soustavě. Statické charakteristiky jsem měřil v obou případech v rozsahu 0–10 V, přičemž krok mezi jednotlivými ustálenými stavy jsem zvolil 1 V (obr. 1.10).



Obr. 1.10 – Průběh měření statické charakteristiky žárovky při nulovém průtoku vzduchu

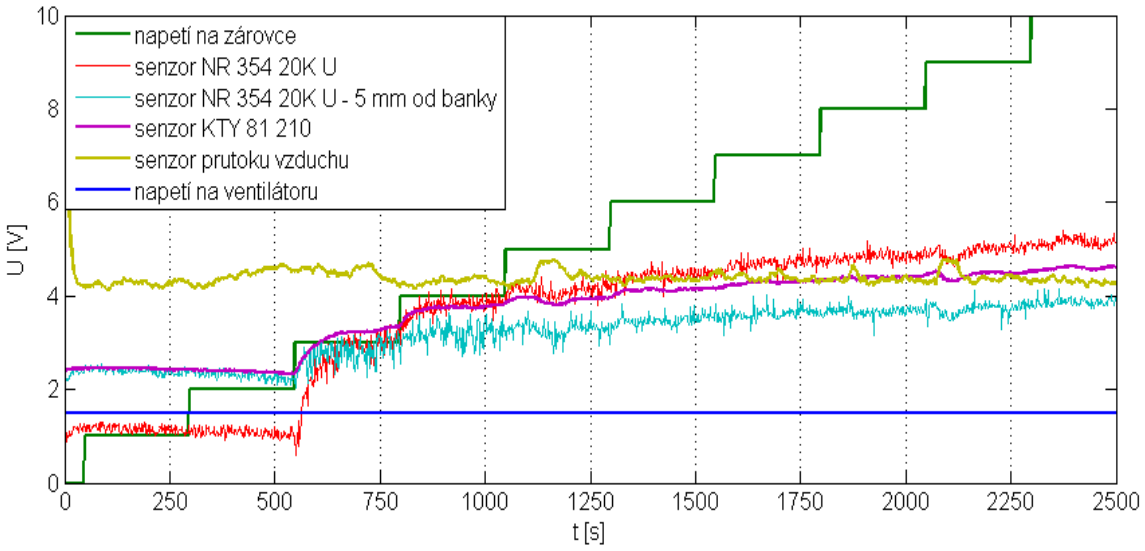
Při měření statické charakteristiky žárovky je z grafu patrné, že výstupní napětí ze senzorů teploty nemá takový charakter, že by se ustálilo na určité hodnotě, ale má tendenci stále růst, žárovka má tzv. drift, neustále je výstupní hodnota unášena. Je to způsobeno tím, že vzduch v okolí žárovky se otepluje a nedochází k proudění vzduchu, proto má žárovka v uzavřeném prostředí daný charakter. V dalších měřeních jsem vyzkoušel, jaký vliv budou mít otáčky ventilátoru na tento jev u žárovky. V grafu je také možné vidět, že některá čidla jsou velmi zašuměná, proto je důležité vybrat nejlepší možné čidlo pro pozdější identifikaci přenosové funkce systému. Z naměřených dat jsem vybral, kvůli zašumění, posledních 10 hodnot před změnou vstupního napětí, ze kterých jsem vypočítal průměr. Po této úpravě jsem následně vykreslil výsledný průběh statické charakteristiky žárovky (obr. 1.11).



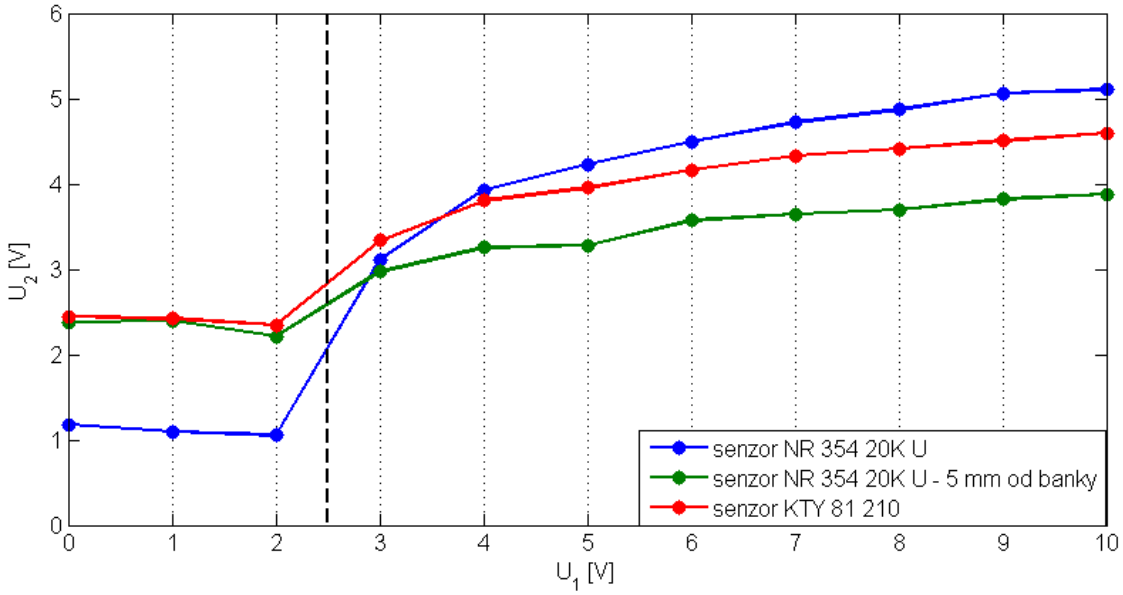
Obr. 1.11 – Statická charakteristika žárovky při nulových otáčkách ventilátoru

Všechny senzory mají téměř shodné průběhy, pouze se liší zesílením v jednotlivých bodech a také se liší v počátcích, protože jsou ovlivněny teplotou okolního prostředí a každý senzor na teplotu reaguje jinak. U žárovky jsem vybral vhodný pracovní bod při 2,5 V na vstupu, a jak lze vidět v grafu, záleží na zvoleném senzoru, podle kterého se bude měnit žádaná hodnota. Zvolený pracovní bod jsem vybral v lineární části statické charakteristiky, protože v této části se při řízení systému nejvíce projeví změna napětí na vstupu.

Následně jsem provedl několik měření, kde jsem měřil statické charakteristiky při různých otáčkách ventilátoru, pro ukázku jsem vybral měření, kdy na ventilátoru jsem nastavil vstupní napětí 1,5 V. U statické charakteristiky při nulových otáčkách jsem mohl naměřit maximální napětí 10 V ze senzorů teploty, ale u takto nastaveného ventilátoru dosahovalo maximální napětí na výstupu senzorů pouze 5 V a méně, lze to vidět na získané statické charakteristice (obr. 1.13). Charakter žárovky, který je dán driftem, se při otáčkách ventilátoru výrazně zmenšil (obr. 1.12), protože vzduch kolem žárovky proudí a nemá tak výrazný vliv na měřenou teplotu.



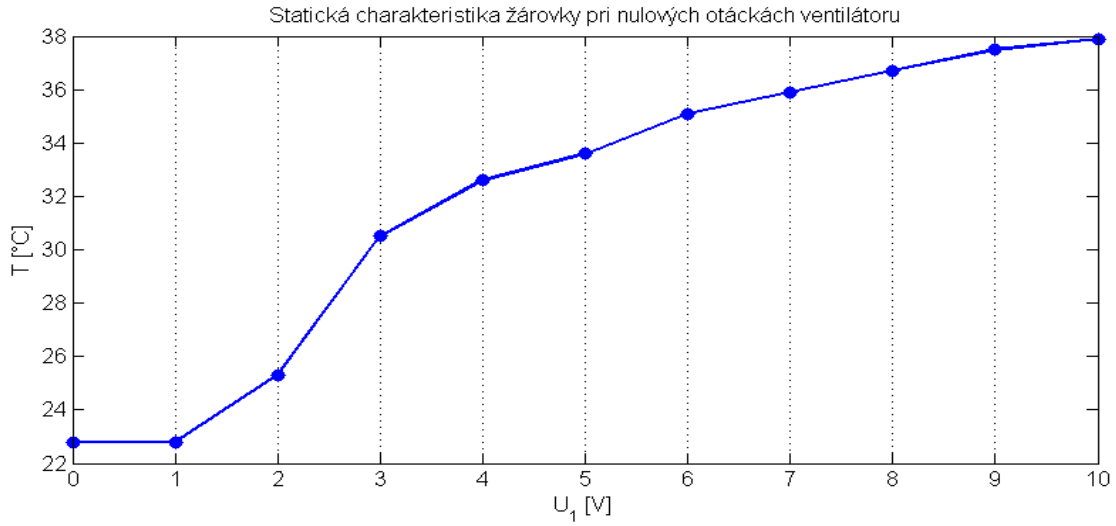
Obr. 1.12 – Průběh měření statické charakteristiky žárovky při 1,5 V na ventilátoru



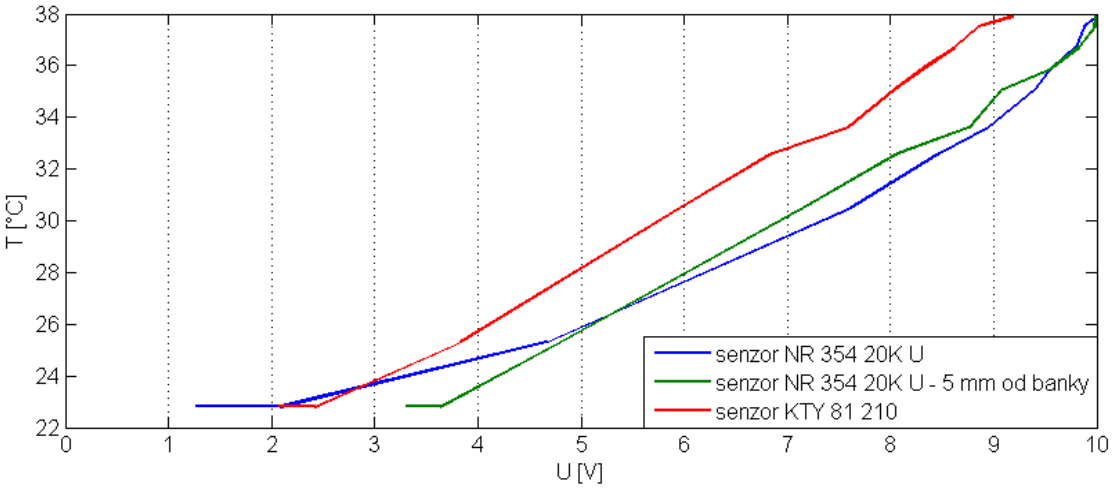
Obr. 1.13 – Statická charakteristika žárovky při 1,5 V na vstupu ventilátoru

Ze statických charakteristik žárovky je zřejmé, že zvolený pracovní bod při 2,5 V na vstupu žárovky se i při zvýšeném průtoku vzduchu udržuje v lineární části daných statických charakteristik, pouze se snižuje maximální dosažitelná teplota na žárovce. Předcházející statické charakteristiky jsou závislosti napětí na vstupu žárovky a výstupního napětí získaného ze senzorů teploty u žárovky. Proto jsem provedl další měření statické charakteristiky pomocí NTC termistoru připojeného na digitální displej *XT11S*, který umožňuje zobrazit přímo teplotu v rozsahu 0–60 °C. Tímto měřením jsem získal statickou charakteristiku (obr. 1.14), která zobrazuje závislost skutečné teploty a vstupního napětí žárovky. Tímto měřením mohu s jistou tolerancí porovnat, jaká hodnota teploty odpovídá výstupnímu napětí ze senzorů teploty

uvnitř soustavy (obr. 1.15). Z převodní charakteristiky je na první pohled zřejmé, že je téměř lineární a záleží na použitém senzoru, kterému odpovídá jiná teplota. Ze získané statické charakteristiky (obr. 1.14) je počátek průběhu ovlivněný teplotou okolního prostředí, kde teplota okolí byla v době provádění měření přibližně 22 °C, a dále z grafu vyplývá, že maximální teplota, kterou může žárovka dosáhnout při nulových otáčkách ventilátoru, je přibližně 38 °C.

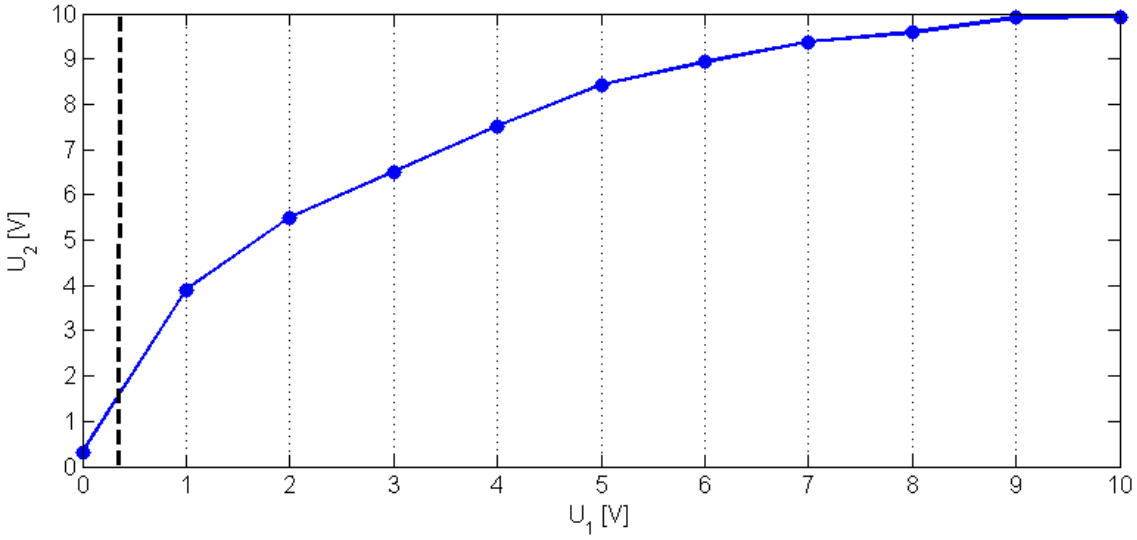


Obr. 1.14 – Závislost vstupního napětí žárovky na skutečné teplotě



Obr. 1.15 – Převodní charakteristika žárovky

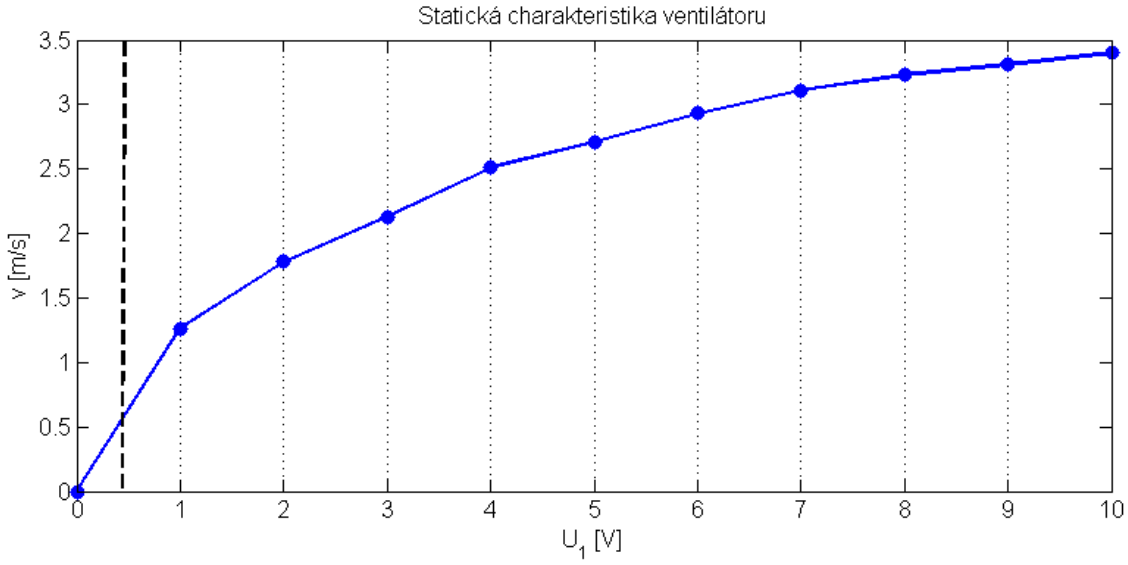
Při měření statické charakteristiky ventilátoru jsem se řídil stejným postupem jako při měření statické charakteristiky žárovky. Měřil jsem ve stejném napěťovém rozsahu 0–10 V, s krokem mezi ustálenými stavy 1 V. Po zpracování naměřených dat jsem získal výslednou statickou charakteristiku ventilátoru (obr. 1.16).



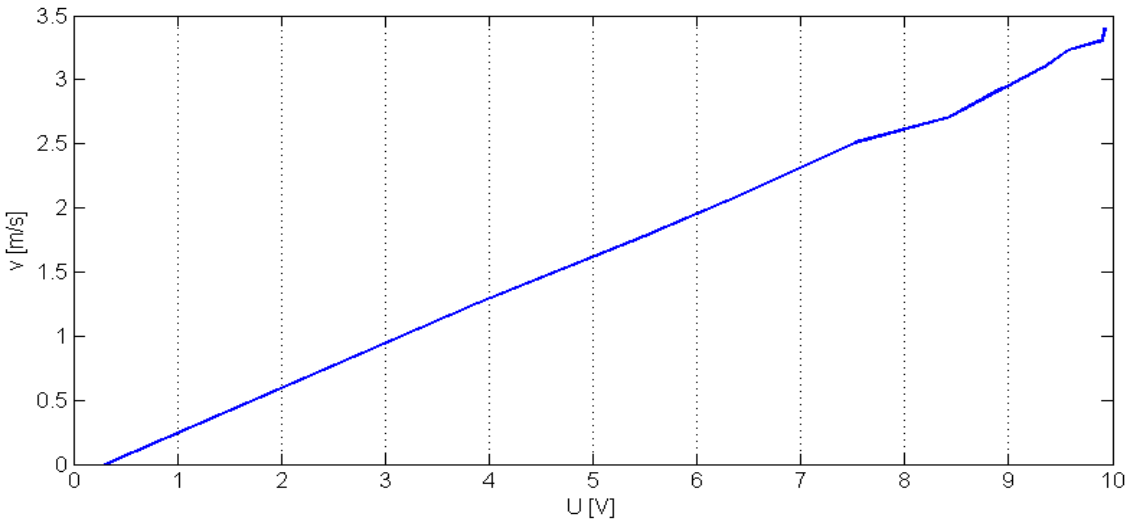
Obr. 1.16 – Statická charakteristika ventilátoru

U ventilátoru optimální pracovní bod vychází ze statické charakteristiky přibližně ve 4 V, ale vhodný pracovní bod jsem zvolil přibližně při 0,5 V na vstupu ventilátoru, protože při regulaci jsem zjistil, že ventilátor nebude možné provozovat na tak vysokých otáčkách, které by byly při optimálním pracovním bodě. Toto omezení vyplývá z křížové vazby mezi ventilátorem a žárovkou, a proto jsem zpětně přehodnotil volbu pracovního bodu, kterému odpovídá přibližně žádaná hodnota 1,5 V.

Následně jsem pomocí lopatkového anemometru vyrobeného firmou Schiltnecht, naměřil přímo rychlost proudění vzduchu. Sonda byla spojena s převodníkem *ALMEMO 8390-2*, který převádí signály ze sondy a zobrazuje přímou hodnotu rychlosti proudění vzduchu v m/s. Tímto provedeným měřením jsem získal statickou charakteristiku (obr. 1.17), kde je vynesena závislost skutečné rychlosti proudění vzduchu na vstupním napětí ventilátoru. Poté jsem ještě vykreslil převodní charakteristiku, která vypovídá o závislosti výstupního napětí ze senzoru průtoku vzduchu na rychlosti proudění vzduchu (obr. 1.18).



Obr. 1.17 – Závislost vstupního napětí ventilátoru na rychlosti proudění vzduchu



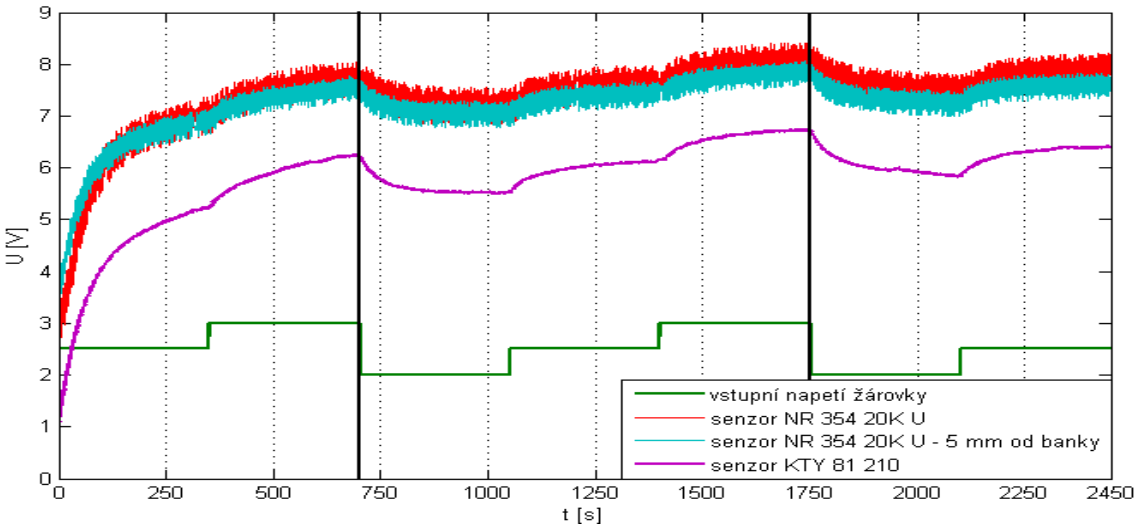
Obr. 1.18 – Převodní charakteristika ventilátoru

Při porovnání statických charakteristik ventilátoru jsou průběhy téměř shodné, maximální rychlost průtoku vzduchu, odpovídající vstupnímu napětí ventilátoru 10 V, je téměř 3,5 m/s. V okolí pracovního bodu se rychlost proudění vzduchu pohybuje kolem 2,5 m/s. Získaná převodní charakteristika je téměř v celém rozsahu lineární. S určitou tolerancí tak mohu přiřadit, jaká rychlost proudění vzduchu odpovídá měřenému výstupnímu napětí pomocí senzoru průtoku vzduchu v soustavě.

1.2.3 Dynamické vlastnosti

Druhou částí identifikace je získání popisu systému ve formě dynamických vlastností, kde tyto vlastnosti vyjadřují, jak se systém chová při přechodu mezi jednotlivými ustálenými stavy. Dynamické vlastnosti systému mohou být reprezentovány různými způsoby, např. diferenciální rovnicí, přenosem nebo přechodovou funkcí. Pro získání všech přenosů jsem musel realizovat příslušná měření, které odpovídají definici dynamických vlastností systému.

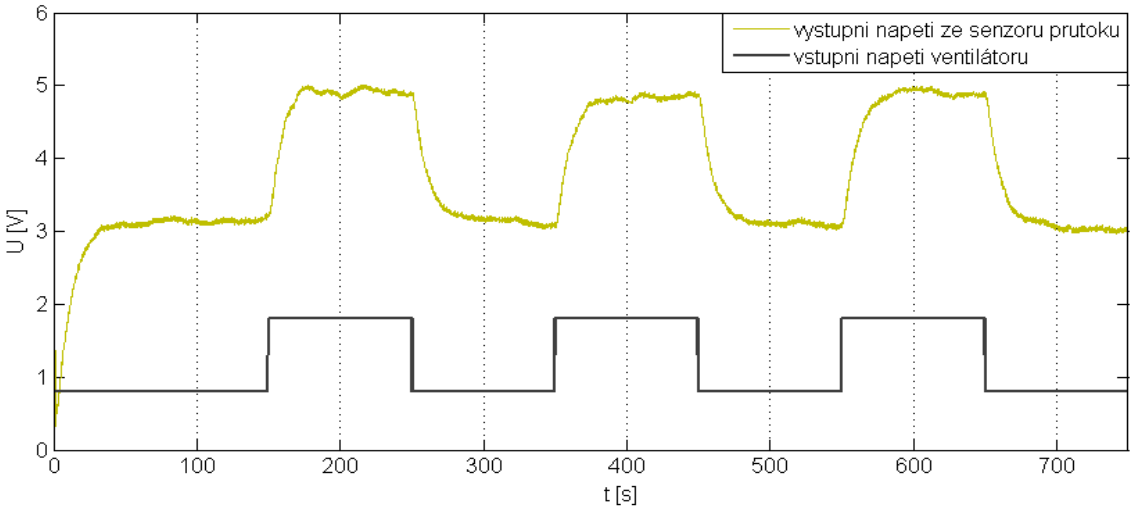
Nejdříve jsem provedl měření dynamiky u systému žárovky (obr. 1.19) tak, že jsem na žárovce nastavil hodnotu pracovního bodu, respektive vstupní napětí žárovky na 2,5 V, a v okolí tohoto bodu jsem prováděl malé skoky. Získaná data jsem musel nejdříve upravit do takové podoby, abych je mohl později využít k identifikaci a získání přenosu. Protože systém žárovky v počátku měření přechází z nedefinovaného stavu do pracovního bodu, musel jsem provést linearizaci v pracovním bodě, která spočívá v odstranění části měření v počátku, kdy systém přechází do pracovního bodu a posunu naměřených dat do nuly. Další problém, který u žárovky nastává, je dříve zmiňovaný drift, proto jsem pro identifikaci žárovky vybral pouze užitečná data v rozsahu 2–3 V vstupního napětí, kdy provádím skoky směrem nahoru.



Obr. 1.19 – Průběh měření dynamických vlastností žárovky při nulovém průtoku vzduchu

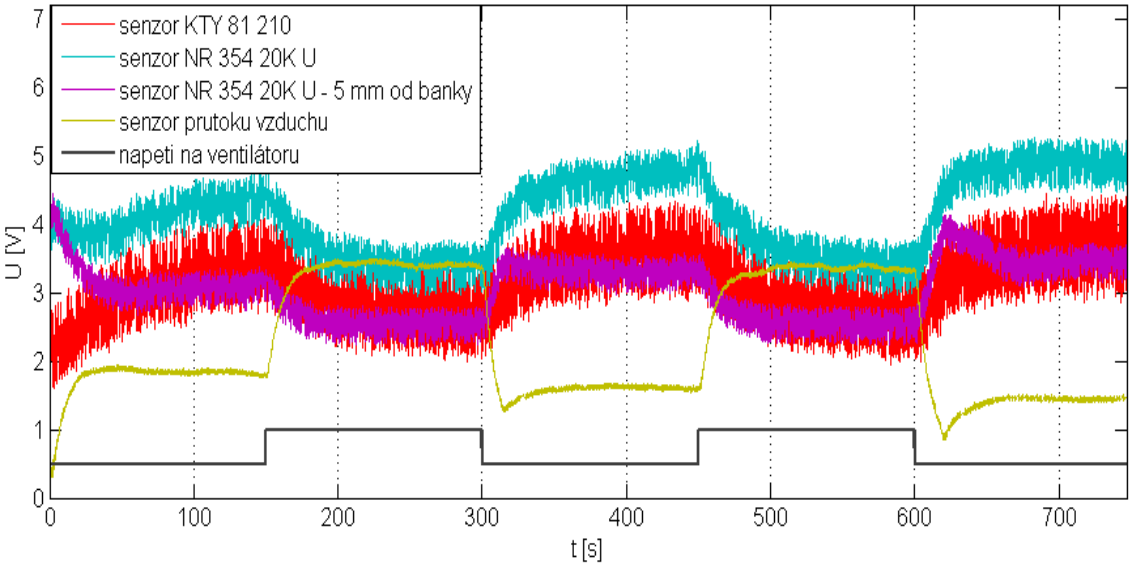
U ventilátoru a křížové vazby se mi nepodařila identifikace na první pokus. Tuto skutečnost jsem ale zjistil až ve fázi, kdy jsem porovnával simulace a reálné odezvy při regulaci. Proto jsem zpětně naměřil identifikaci ventilátoru a křížové vazby, kde jsem tyto dvě měření provedl zároveň a následně jsem získal žádané přenosy, které už jsou správné. Dynamiku ventilátoru (obr. 1.20) jsem měřil v okolí pracovního bodu,

který jsem zvolil 1 V na vstupu, kde v okolí tohoto bodu jsem prováděl malé skoky. Z naměřených dat mohu považovat téměř všechna data za užitečná, protože u žárovky toto možné nebylo, a pouze je nutné provést linearizaci v daném bodě, kde na začátku měření dochází k přechodu z nedefinovaného stavu do pracovního bodu.



Obr. 1.20 - Průběh měření dynamických vlastností ventilátoru

Posledním důležitým měřením u dynamických vlastností soustavy je měření křížové vazby mezi ventilátorem a žárovkou (obr. 1.21). Toto měření jsem provedl tak, že jsem na žárovce nastavil hodnotu pracovního bodu, která byla po celou dobu měření konstantní, a prováděl jsem malé skoky vstupního napětí na ventilátoru. Naměřená data jsem následně stejnými postupy upravil, jako v předešlých měřeních. Pro pozdější využití měření v identifikaci jsem vybral pouze oblast prvního zvýšení otáček ventilátoru.



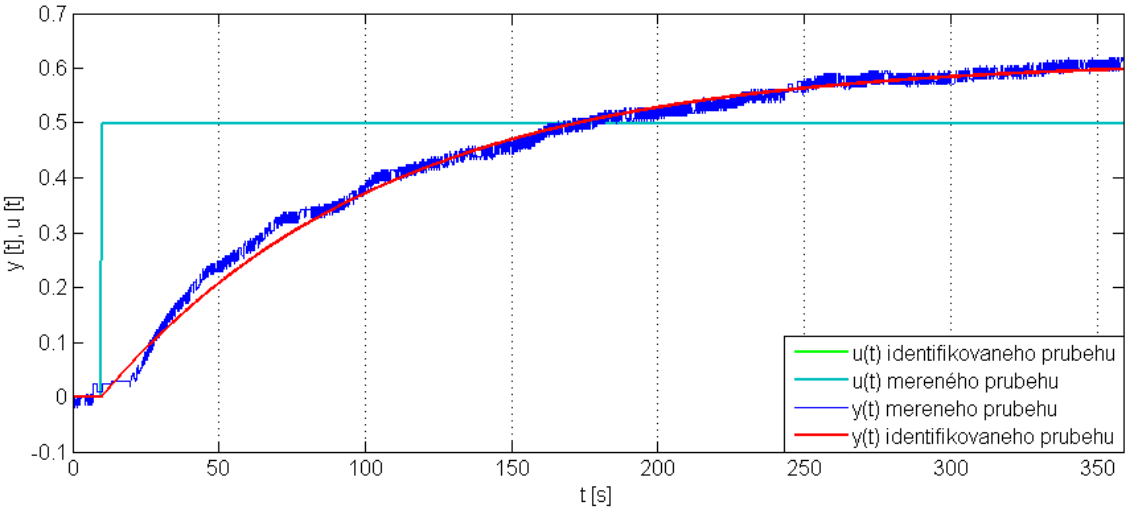
Obr. 1.21 – Průběh měření křížové vazby

1.2.4 Verifikace naměřených dat

Verifikace je proces, při kterém se ověřuje shoda mezi reálným systémem a identifikovaným systémem. Z naměřených dat, která jsem podle potřeby upravil, mohu pomocí nástroje, který je standardní součástí Matlabu, určit přenosy systémů. V Matlabu se spouští příkazem **ident**, a je součástí panelu nástrojů *Identification Toolbox*. Jinou možností, jak získat přenos systému, je využití optimalizačního programu, jehož základem bude v principu funkce fminsearch, která hledá lokální minimum funkce v okolí zadaného bodu nebo v určitém rozmezí. Pro potřeby získání přenosu jsem využil identifikační nástroj, protože jeho obsluha je snadná a realizuje vše, co by bylo potřeba naprogramovat.

Pro systém žárovky jsem pomocí identifikačního nástroje vypočítal funkci, která se v porovnání s naměřeným průběhem shoduje na 91,27 % (obr. 1.22). Vybraný naměřený průběh pro identifikaci je přechod z 2,5 V do 3 V ve vymezeném rozmezí (viz obr. 1.19). Z naměřených a porovnaných průběhů lze vidět, že akční zásahy se překrývají a identifikovaný přenos je správný. Matematickým vyjádřením této shody je výsledný přenos 1. řádu.

$$F_1(s) = \frac{1{,}23}{96{,}89s+1} \tag{1.4}$$

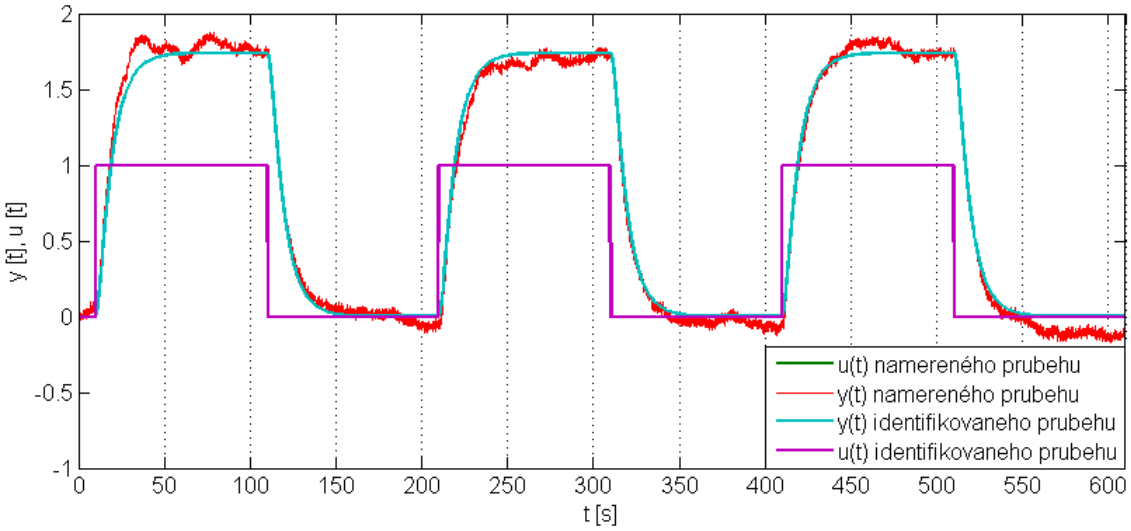


Obr. 1.22 – Porovnání naměřených průběhů a identifikovaných u žárovky

Pro systém ventilátoru jsem identifikačním nástrojem zjistil funkci, která se v porovnání s naměřenými daty shoduje s pravděpodobností 92,1 % (obr. 1.23). Při porovnání průběhů u ventilátoru se opět akční zásahy překrývají, proto je

identifikace správná. Matematické vyjádření ve formě přenosu vychází jako přenos 2. řádu.

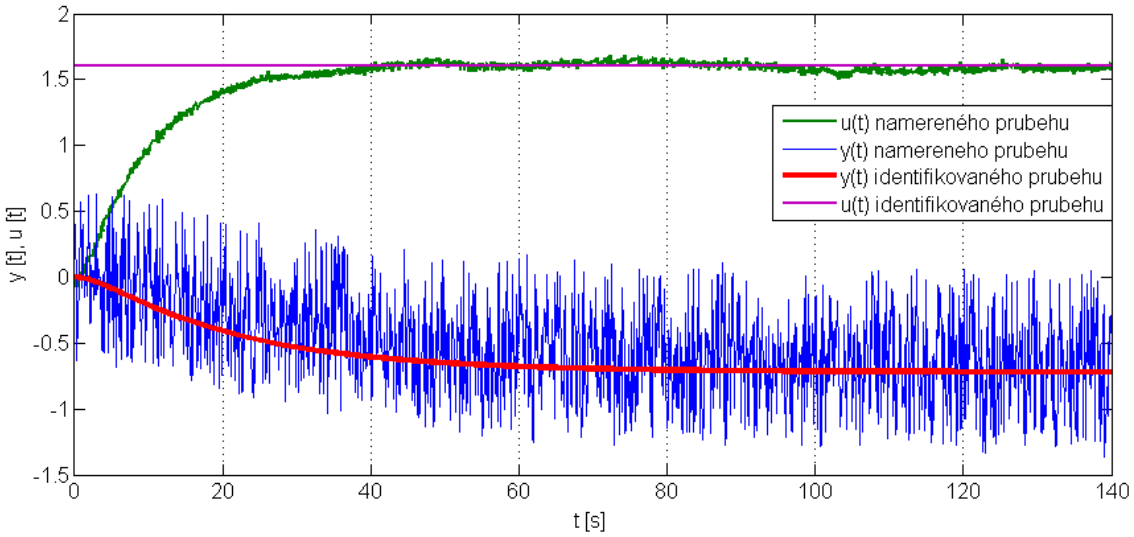
$$F_2(s) = \frac{1{,}74}{(8{,}10s+1)(2{,}01s+1)} = \frac{1{,}74}{16{,}25s^2+10{,}11s+1} \tag{1.5}$$



Obr. 1.23 –Porovnání naměřených průběhů a identifikovaných u ventilátoru

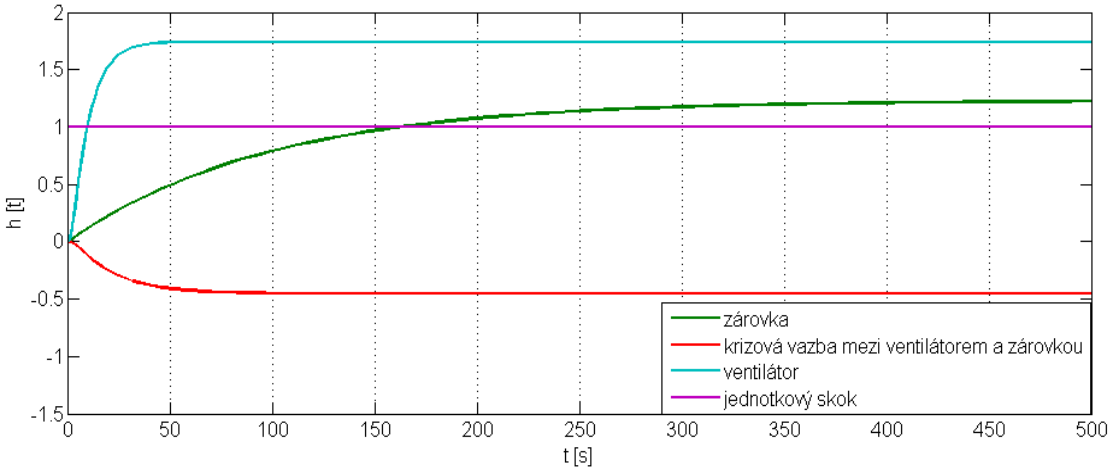
Poslední přenos, který jsem potřeboval zjistit, byl přenos křížové vazby mezi ventilátorem a žárovkou. Pravděpodobnostní shodu mezi naměřenými daty a identifikovanými vyhodnotil identifikační nástroj pouze na 50 %, protože výstup senzoru teploty je poměrně zašuměný, ale při použití průměrovacího filtru, se pravděpodobnost zvýší. Průměrovací filtr jsem realizoval tak, že z naměřených dat se pro každých 10 hodnot vypočítal průměr a získala se tak nová hodnota. Celkový počet naměřených bodů se snížil 10x. Z toho vyplývá, že procentuální shoda udává pouze orientační hodnotu, ale při pohledu na porovnané průběhy (obr. 1.24) je zřejmé, že identifikace je provedená správně. Z naměřeného průběhu jsem pro účely identifikace využil pouze část průběhu, kde dochází k prvnímu zvýšení otáček ventilátoru a senzor, který jsem použil pro identifikaci, byl *NR 354 20K U*, který se nachází 5 mm od baňky žárovky. Při porovnání průběhů křížové vazby se akční zásahy v počátku nepřekrývají, protože naměřený akční zásah je zde výstupem ze senzoru průtoku vzduchu a nemá tak rychlý přechod mezi ustálenými stavy jako simulovaný akční zásah. Křížová vazba je tak závislost regulované veličiny, kterou je teplota na žárovce a akčního zásahu, kterým je výstup ze senzoru průtoku vzduchu. Matematické vyjádření přenosu křížové vazby vychází ve formě 2. řádu.

$$F_{12}(s) = -\frac{0{,}45}{(20{,}04s+1)(2{,}93s+1)} = -\frac{0{,}45}{58{,}73s^2+22{,}97s+1} \tag{1.6}$$



Obr. 1.24 – Porovnání naměřených a simulovaných průběhů křížové vazby

Na závěr jsem vykreslil přechodové charakteristiky (obr. 1.25), kde podle známé definice je přechodová charakteristika odezva na jednotkový skok. Z vykreslených průběhů je na první pohled zřejmé, že ventilátor má velmi rychlé reakce, do ustáleného stavu se dostane za krátkou dobu, ale žárovka má velkou setrvačnost, než dosáhne ustáleného stavu, uběhne poměrně dlouhá doba.



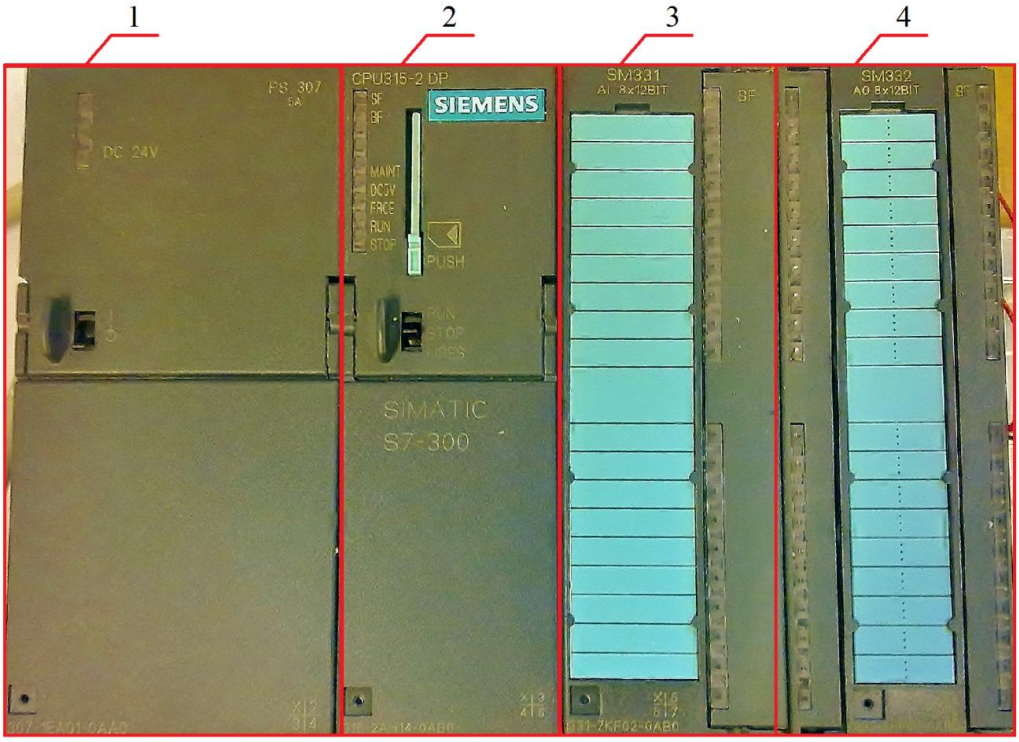
Obr. 1.25 – Přechodové funkce soustavy

2 Realizace a struktury řízení

2.1 Popis PLC

Pro řízení reálné soustavy jsem využíval PLC od firmy Siemens, které pochází z modulové řady *S7-300*. Toto PLC se řadí mezi modulární automaty, a to znamená, že funkci PLC lze rozšířit různými přídavnými moduly tak, aby se PLC co nejvíce přizpůsobilo dané aplikaci. Celkové řízení vícerozměrového systému pomocí PLC může být realizované dvěma způsoby, a to jako centralizované nebo decentralizované řízení. Nejdůležitější částí PLC je hlavní řídicí jednotka CPU, kde u řady S7-300 může mít různá provedení. PLC může obsahovat standardní CPU, které pouze slouží pro vykonávání programu a tento typ CPU je v praxi nejpoužívanější, dále může obsahovat kompaktní CPU, které je doplněné o digitální nebo analogové vstupy a výstupy, další možností jsou bezpečnostní CPU nebo technologická CPU, která obsahují předem naprogramované různé funkce pro řízení polohy a pohybu.

Pro řízení vícerozměrové soustavy jsem využíval PLC sestavu (obr. 2.1), která se skládá z napájecího zdroje (1), který je připojený na síťové napětí *230 V* a PLC je ze zdroje napájeno stejnosměrným napětím *24 V*. Hlavním modulem, který sestavu řídí, je CPU *315-2 DP* (2), které disponuje integrovanou pamětí o velikosti **128 kB**, která umožňuje zapsat přibližně **42** tisíc instrukcí. Pro snadnější zálohování konfigurace CPU, případně programu uloženého v paměti, obsahuje CPU slot pro *MMC* karty. Ke komunikaci s CPU se využívá komunikačního rozhraní *MPI* nebo *Profibus*. Protože standardní počítač není vybaven rozhraním *MPI*, tak bych nemohl komunikovat s PLC, ale existují převodníky mezi *MPI* a komunikačním rozhraním PC. Standardně jsou to převodníky z *MPI* na *RS-232* nebo *USB*, kde k dispozici jsem měl variantu s *USB*. Rozšiřující moduly, které jsem pro účely řízení potřeboval, byly moduly analogových vstupů *SM331-7KF02-0AB0* (3) a analogových výstupů *SM332-5HF00-0AB0* (4), které uvnitř obsahují A/D převodník a D/A převodník. Moduly vstupů a výstupů disponují počtem **8** kanálů, které mají rozlišení **12** bitů.



Obr. 2.1 – PLC Simatic S7-300

Modul analogových vstupů se skládá ze dvou částí, z hlavního modulu *SM331* a dále se skládá z přídavného modulu, který udává, v jakém rozsahu a v jakých jednotkách jsou přivedené hodnoty na vstupech. Tento přídavný modul je zasazen do hlavního modulu, a jeho orientací se určí jedna ze čtyř možností typu měření (tab. 1), kde je potřeba nastavit správnou orientaci, aby nedocházelo ke čtení nesmyslných hodnot. Modul vstupů je tvořen tak, že každá dvojice vstupů tvoří jednu skupinu kanálu a celkem jsou tak obsaženy v modulu 4 skupiny po 2 kanálech. Ke každé skupině přísluší jeden přídavný modul, který určuje rozsah měření pro dané dva vstupy. Pro potřeby řízení reálné soustavy jsem využíval pouze jednu skupinu kanálu, respektive 2 vstupy a přídavný modul jsem nastavil pro variantu B (viz tab. 1), protože jsem měřil hodnoty v rozsahu 0-10 V, kde při programování rozsah odpovídá 0–27 648 hodnot.

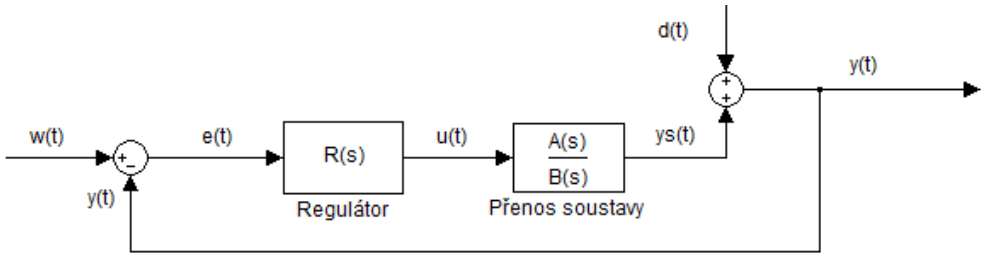
Tab. 1 – Možnosti nastavení modulu měřicího rozsahu

Orientace přídavného modulu	Metoda měření	Rozsah měření
A	Termoelektrické nebo	± 1000 mV
B	Napěťové	± 10 V
C	4-vodičové měření proudu	4 až 20 mA
D	2-vodičové měření proudu	4 až 20 mA

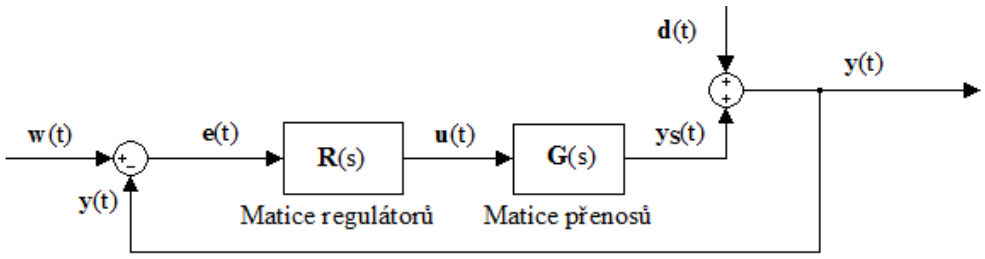
2.2 Návrh regulátorů

Návrh regulátorů pro řízení reálné soustavy se provádí z toho důvodu, aby se dosáhlo zlepšení odezvy systému. Funkce regulátoru ve zpětnovazební regulační smyčce je udržovat minimální regulační odchylku, neboli minimální rozdíl žádané hodnoty a regulované veličiny. Regulátor vytváří takové akční zásahy pro systém, které umožňují zrychlit náběh systému na žádanou hodnotu, vyplývá to z toho, že se regulátor snaží udržet regulační odchylku co nejmenší. Jakmile dosáhne výstup systému ustáleného stavu na žádané hodnotě, regulátor bude vytvářet odpovídající akční zásahy tak, aby výstup systému udržel na žádané hodnotě.

Pro návrh regulátoru je několik obecných postupů, které udávají vztahy pro výpočet jednotlivých parametrů regulátoru. Návrhové metody tak usnadňují proces získávání parametrů regulátoru, ale ne vždy je využití návrhových metod nejlepší volba, protože různé aplikace vyžadují různé postupy návrhu. Mezi nejpoužívanější metody patří pravidla Zieglera a Nicholse, Cohen Coonova metoda, metoda souhrnné časové konstanty, nazývaná také jako Kuhnova metoda, nebo experimentální ruční nastavení parametrů. U SISO soustavy je návrh regulátoru nejjednodušší, protože zpětnovazební regulační struktura obsahuje pouze jediný regulátor (obr. 2.2). Naopak je tomu u MIMO soustavy, kde návrh regulátoru je složitější, protože soustava se skládá z více systému, a tomu odpovídá i nutnost realizace více regulátorů. Regulátor v dané soustavě tak může být realizovaný jako matice regulátorů (obr. 2.3), nebo může být realizovaný větším počtem samostatných regulátorů, kde tento počet odpovídá počtu systémů.



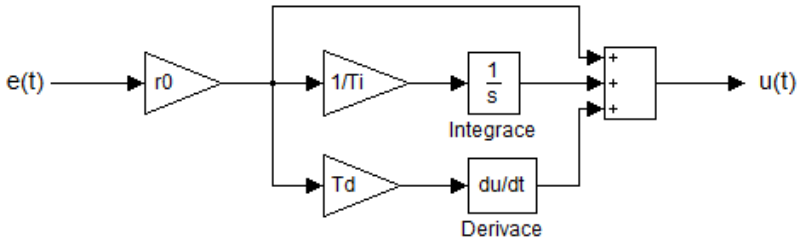
Obr. 2.2 – Obecná struktura zpětnovazební SISO regulační smyčky



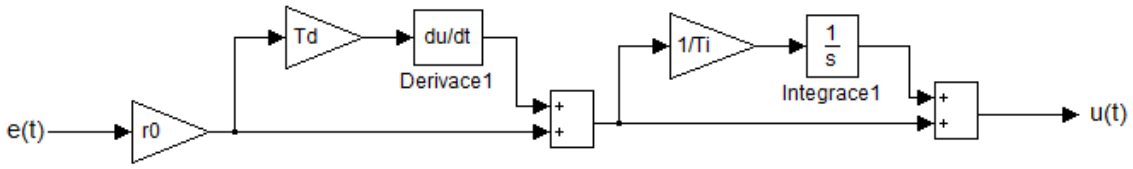
Obr. 2.3 – Obecná struktura zpětnovazební MIMO regulační smyčky

Regulátory se skládají obecně z proporcionální (P), integrační (I) a derivační (D) složky, kde jejich kombinací se vytvoří regulátory pro optimální řízení. P složka realizuje pouze zesílení regulátoru, I složka je zařazena z důvodu nulové regulační odchylky, ale může způsobovat při nesprávném nastavení nestabilitu systému a D složka ovlivňuje celkovou dynamiku systému, respektive ovlivňuje rychlost náběhu systému. Nejčastěji se navrhují PID regulátory, které tvoří všechny složky a umožňují kvalitní regulaci, nebo je možné využít pouze kombinaci PI regulátoru. Struktura regulátorů je častěji paralelní (obr. 2.4), kde popis v časové oblasti je dán vztahem (2.1), ale může se vyskytovat i struktura sériová (obr. 2.5).

$$u(t) = r_0 \cdot \left[e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau) d\tau + T_d \cdot \frac{de(t)}{dt} \right] \tag{2.1}$$



Obr. 2.4 – Paralelní struktura PID



Obr. 2.5 – Sériová struktura PID

Z možných struktur regulátorů jsem využíval paralelní strukturu, protože všechny složky regulátoru mohu nastavovat nezávisle. Pro řízení reálné soustavy jsem vyzkoušel několik zmíněných návrhových metod, přičemž metoda Zieglera a Nicholse a metoda Cohen Coonova byly pro danou soustavu nepoužitelné, protože vypočítané parametry způsobovaly v simulacích velké překmity. Lepších výsledků jsem dosáhl při použití Kuhnovy metody, kde celkový regulační pochod byl přijatelný. Tato metoda má definované pro různé varianty regulátorů výpočetní vztahy, kterými se vypočítají příslušné složky regulátoru. Základem pro výpočet parametrů je určení hodnoty zesílení a souhrnné časové konstanty z přenosu soustavy. Pro přenos systému, který je dán obecným vztahem (2.2), se provede snadný výpočet souhrnné časové konstanty (2.3).

Ze získaných hodnot lze dosadit do definovaných vzorců (tab. 2) a podle typu regulátorů se vypočítají příslušné parametry pro nastavení.

$$G(s) = K \cdot \frac{(T_{1N} \cdot s+1)(T_{2N} \cdot s+1) \cdots (T_{mN} \cdot s+1)}{(T_1 \cdot s+1)(T_2 \cdot s+1) \cdots (T_n \cdot s+1)} \cdot e^{T_D \cdot s} \tag{2.2}$$

$$T_\Sigma = T_1 + T_2 + \cdots + T_n - T_{1N} - T_{2N} - \cdots - T_{mN} + T_D \tag{2.3}$$

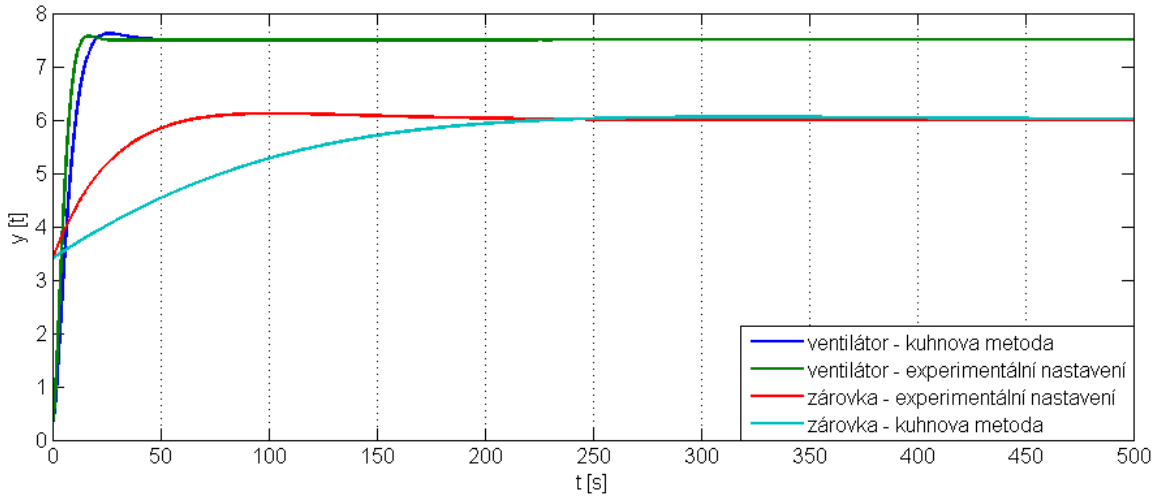
Tab. 2 – Definované vztahy pro metodu souhrnné časové konstanty

Typ regulátoru	r_0	T_i [s]	T_d [s]
P	1/K		
PD	1/K		$0{,}33 \cdot T_\Sigma$
PI	1/K	$0{,}7 \cdot T_\Sigma$	
PID	2/K	$0{,}8 \cdot T_\Sigma$	$0{,}194 \cdot T_\Sigma$

Poslední možností, kterou jsem využil, je určení parametrů regulátoru experimentální metodou, která umožňuje nastavit regulátor pro takovou regulaci, která bude co možná nejlepší. Pro řízení obou systémů jsem si vybral variantu PI regulátorů, která se později ukázala jako dostačující pro regulaci. V následující tabulce (tab. 3) jsem uvedl vypočítané parametry regulátoru pomocí Kuhnovy metody a pomocí experimentálního nastavení. Z tabulky je na první pohled patrné, že integrační časové konstanty u použitých nastavení se téměř shodují, pouze se výrazně změnilo zesílení regulátoru. Pro výběr nejvhodnější metody jsem provedl grafické porovnání navržených regulátorů (obr. 2.6). Z porovnání je zřejmé, že experimentální nastavení je vhodnější než Kuhnova metoda, proto jsem nadále pracoval s experimentálně nastavenými regulátory, které jsem následně také aplikoval do PLC. U ventilátoru rozdíl mezi metodami není příliš patrný, ale experimentální nastavení je rychlejší, naopak u žárovky je nastavení znatelné více, kde použitý regulátor má rychlejší dynamiku. Protože v reálné soustavě měřené veličiny nejsou zcela nulové, proto jsem uvažoval při porovnání regulací tento offset v počátku simulace.

Tab. 3 – Navržené parametry regulátorů

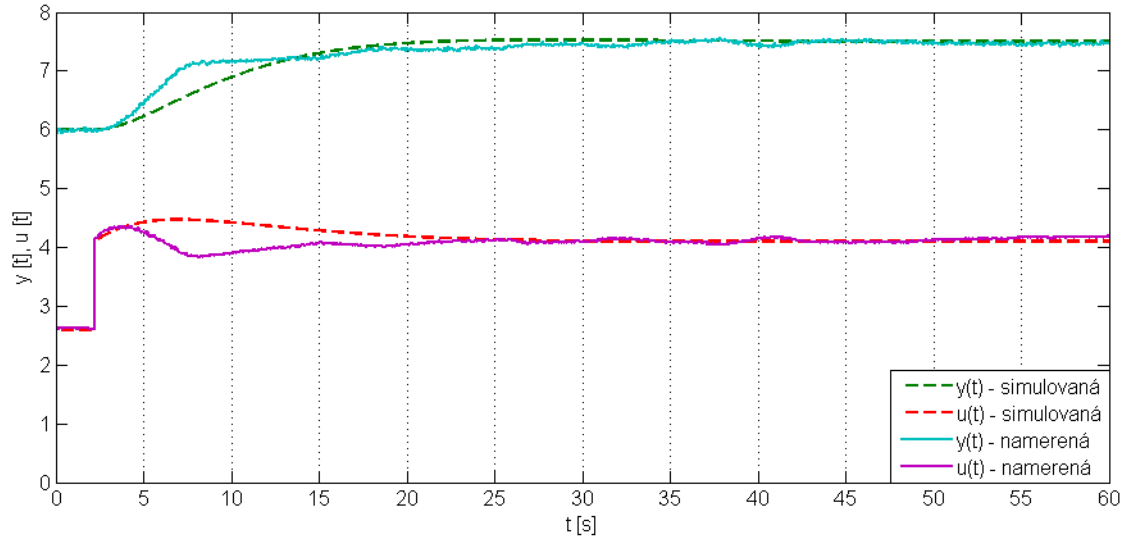
Kuhnova metoda					
Systém	Zesílení	T_Σ [s]	Typ regulátoru	r_0	T_i [s]
Žárovka	1,23	96,89	PI	0,81	67,82
Ventilátor	1,74	10,11	PI	0,57	7,08
Experimentální nastavení					
Systém				r_0	T_i [s]
Žárovka				3,2	60
Ventilátor				1	7



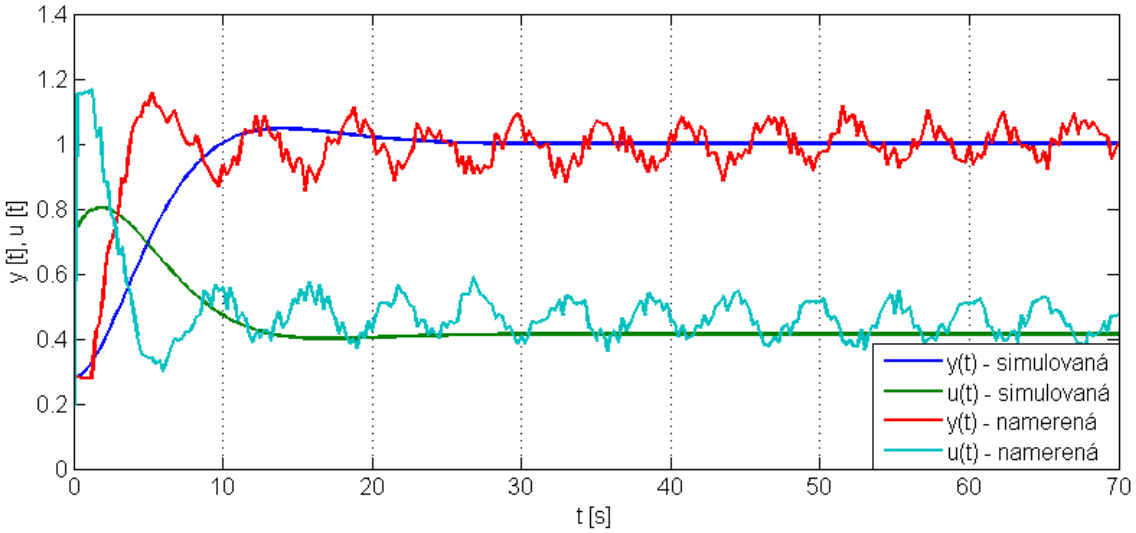
Obr. 2.6 – Graf porovnání návrhových metod

Pro ověření odezvy simulací a odezvy reálných systémů jsem aplikoval nastavené regulátory do PLC a nastavil jsem hodnoty pracovních bodů systémů. Poté jsem pomocí multifunkční karty a Matlabu naměřil odezvy reálných systémů a následně jsem porovnal regulované veličiny a akční zásahy se simulovanými (obr. 2.7) a (obr. 2.9). Z porovnaných průběhů je vidět, že v některých částech dochází k menším odchylkám mezi naměřenými a simulovanými průběhy, může to být způsobené nepřesnostmi při identifikaci, protože nikdy není zaručena ideální identifikace. U ventilátoru ještě dochází k tomu, že se může měnit zesílení přenosu v rozmezí 1-1,8. Při měření průběhu ventilátoru jsem provedl nejdříve skok do okolí optimálního pracovního bodu, v tomto případě skok na hodnotu 6 V, která byla na výstupu senzoru průtoku vzduchu, a poté jsem zvýšil hodnotu na 7,5 V, respektive hodnotu optimálního pracovního bodu. Z toho důvodu nezačínají porovnané průběhy v 0 V, ale počátky měření jsou na hodnotě 6 V pro regulované veličiny.

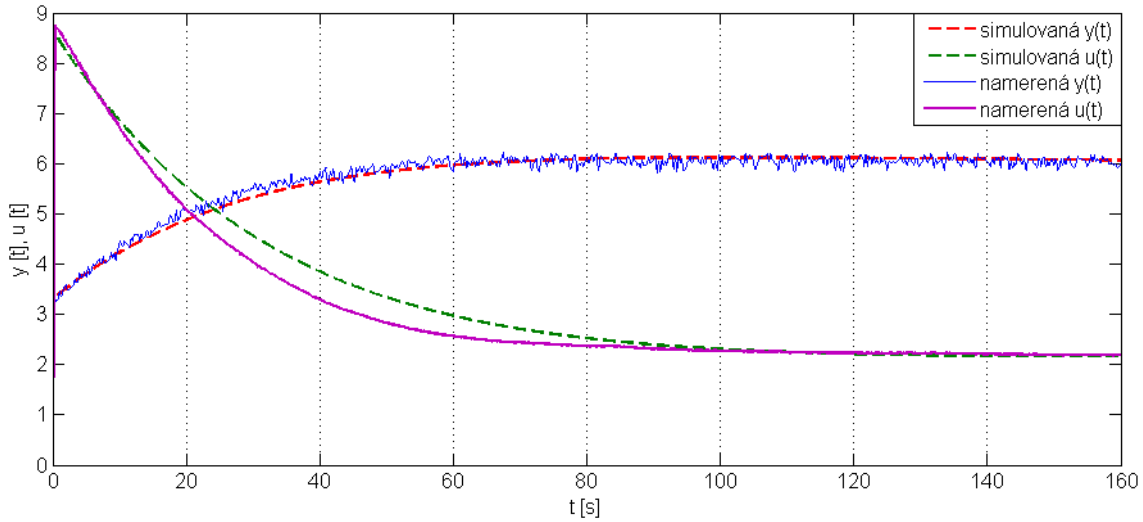
Protože jsem později zjistil, že ventilátor nebude možné regulovat v optimálním pracovním bodě, provedl jsem srovnání simulace navrženého regulátoru pro ventilátor a naměřeného průběhu v pracovním bodě 1 V na vstupu ventilátoru (obr. 2.8). Při porovnání průběhů se v části počátku měření mírně odlišují, protože ventilátor přechází z klidu do pracovního bodu, a tak se simulace v počátku měření mírně odlišuje od naměřeného průběhu. Při regulaci ventilátoru v takto nízkých otáčkách vzniká problém s rozlišením senzoru průtoku vzduchu! Při měření průběhu na žárovce jsem tento postup neprováděl, protože senzory teploty mají trvalý offset v důsledku teploty okolního prostředí a mohl jsem tak přímo porovnat simulované a naměřené průběhy.



Obr. 2.7 - Porovnání regulace u ventilátoru – optimální pracovní bod 7,5 V (žádaná hodnota)



Obr. 2.8 – Porovnání regulace u ventilátoru – skutečný pracovní bod 1 V (žádaná hodnota)



Obr. 2.9 – Porovnání regulace u žárovky

2.3 Programování PLC

Použité PLC Simatic řady S7-300 se programuje pomocí softwaru Step 7 vytvořeného firmou Siemens. Jedná se o specifický program, který umožňuje naprogramovat funkci PLC pomocí několika různých programovacích jazyků, konfigurovat PLC a kontrolovat jeho stav. Step 7 obsahuje programovací jazyk *STL*, kde struktura jazyka je podobná Assembleru, ale příkazy pro programování jsou samozřejmě zcela odlišné. Dále obsahuje programování pomocí *LAD* diagramu, který má jistou podobnost s releovými schématy, kde se vytváří schéma pomocí logických bloků, kde po splnění vstupní podmínky se sepne přiřazený výstup, který nemusí být přímo reálný výstup, ale může se nastavit pouze nějaký paměťový bit nebo příznak. Podobným programovacím postupem jako *LAD* je i *FBD*, který je tvořen bloky obsahující hotové funkce, proto je tato forma programování snadnější než vytváření *LAD* diagramu. V programu Step 7 se nacházejí i vyšší programovací jazyky, jako *SCL*, který má jistou podobnost s jazykem Pascal, některé příkazy v jazyku *SCL* se zdají být téměř stejné jako používané instrukce v jazyku Pascal, ale příkazy plnící stejnou funkci se v *SCL* zapisují odlišně, někdy zcela jinak. Poslední možnost, jak programovat PLC ve Step 7, je využití *Graph*, kde toto je sekvenční programovací jazyk, programuje se pomocí bloků, kde se vykonává program, dokud se nesplní podmínka pro přechod do dalšího bloku.

V programu se pracuje s proměnnými, jejichž typy jsou všeobecně známé, např. proměnná typu *BOOL* (1 bit), *BYTE* (8 bitů), *WORD* (16 bitů), *DWORD* (32 bitů), *INT* (16 bitů) nebo *REAL* (32 bitů). Mezi proměnnými všech typů lze samozřejmě pomocí příslušných instrukcí převádět. Základními bloky, které se vyskytují ve Step 7, jsou organizační bloky (*OB*), funkční bloky (*FB*), bloky funkcí (*FC*) a bloky dat (*DB*). Při programování jsem nejvíce používal proměnné typu *WORD, INT* a *REAL*. Typ *WORD* jsem především využíval při čtení a zápisu hodnot na analogové vstupy a výstupy, protože tyto moduly nemohou pracovat s jinými typy než *WORD, DWORD* a *BYTE*, a protože používaný rozsah modulů je v rozmezí 0-10 V, kterému odpovídá rozsah proměnné 0–27 648. Do tohoto rozsahu je *BYTE* nedostačující, má pouze 256 hodnot, *DWORD* bych mohl využít, ale postačující je typ *WORD*, který má přibližně 32 000 hodnot. Proměnným používaným v programu se mohou přiřadit i symbolická jména (obr. 2.10), takže není nutné si pamatovat adresu proměnné v paměti.

S7 Program(1) (Symbols) -- Dekompozice\SIMATIC 300 Station\CPU315-2 DP(1)

	Status	Symbol	Address	Data type	Comment
1		CONT_C	FB 41	FB 41	Continuous Control
2		Cycle Execution	OB 1	OB 1	
3		Cyclic Interrupt 5	OB 35	OB 35	
6		u1	PQW 274	WORD	Akční zásah pro žárovku
7		u2	PQW 272	WORD	Akční zásah pro ventilátor
8		VAT_1	VAT 1		
9		y1	PIW 262	WORD	Regulovaná velicina žárovky
10		y2	PIW 260	WORD	Regulovaná velicina ventilátoru

Obr. 2.10 – Tabulka symbolických jmen

Při prvním spuštění Step 7 program nabídne vytvoření nového projektu pomocí rychlé nabídky, kde se zjistí několik základních údajů, jako použitý typ CPU a organizační bloky, které bude uživatel využívat. Pro potřeby programování jsem využil dva organizační bloky, první blok *OB1*, který musí být základem každého projektu, protože to je hlavní organizační blok systému, který cyklicky vykonává v nastavené době celý program, a druhý blok který jsem využíval, byl blok *OB35*, který slouží k tomu, že program uvnitř *OB35* se bude vykonávat vždy po nastavené době. To znamená, že pokud by doba byla nastavená na 10 ms, tak po této době se vždy vykoná program uvnitř *OB35*.

Pro realizaci regulace reálného systému jsem využil programovacího jazyka *STL* (obr. 2.12). Velkou výhodou Step 7 je, že obsahuje velké množství předem připravených funkcí a funkčních bloků, které realizují převod proměnných, komunikaci a mnoho dalších, a také obsahuje funkční blok PID regulátoru (*FB41*), který mě nejvíce zajímal a který jsem dále využíval. Struktura předpřipraveného PID regulátoru je paralelní (viz příloha 1). Abych mohl využívat předpřipravený PID regulátor, musel jsem použít organizačního bloku *OB35*, do kterého jsem implementoval PID blok, a kde jsem realizoval celé řízení. Implementace PID bloku do *OB35* není složitá, k vyvolání PID regulátoru slouží jediná instrukce, poté je nutné nastavit všechny parametry funkčního bloku pro správnou funkci regulátoru. Mezi parametry, které se nastavují, patří např. povolení jednotlivých složek PID regulátoru, protože jsem využíval pouze PI regulátor, D složku jsem mohl zakázat, dále se nastavují hodnoty zesílení regulátoru, integrační časová konstanta, kde jsem tyto hodnoty nastavil podle získaných parametrů z experimentální metody, také se nastavuje žádaná hodnota a mnoho parametrů funkčního bloku (viz příloha 1). Pro ukázání jsem uvedl příklad nastavení několika parametrů PID regulátoru (obr. 2.11), ale PID blok obsahuje mnohem více parametrů, nicméně zobrazené patří mezi důležité pro funkci regulátoru.

Na vstup regulátoru jsem přiváděl přímo hodnoty z analogových vstupů a výstup regulátoru jsem opět přiváděl na analogové výstupy. Pro ukládání proměnných slouží datové bloky, které si mohu sám vytvořit, ale pokud chci využívat některé předpřipravené funkční bloky, je nutné jim přiřadit datové bloky, které uloží veškeré nastavení funkčního bloku. Možnost sledování proměnných je v programu pomocí tzv. *VAT* tabulky, kde se vytvoří, pomocí adresy nebo symbolického jména příslušného datového bloku, seznam proměnných, které lze za provozu v PLC sledovat.

```
CALL "CONT_C" , DB41       //volání bloku FB41 a přiřazení datového bloku DB41
  PVPER_ON := TRUE         //povolení vstupu regulátoru
  P_SEL    := TRUE         //povolení P složky
  I_SEL    := TRUE         //povolení I složky
  D_SEL    := FALSE        //zakázání D složky
  CYCLE    := T#5MS        //perioda vzorkování
  SP_INT   := DB41.DBD6    //žádaná hodnota
  PV_PER   := PIW260       //čtení regulované veličiny přímo z analogového vstupu na adrese 260
  GAIN     := 1.0          //zesílení regulátoru
  TI       := T#7S         //inegrační časová konstanta
  TD       := T#0MS        //derivační časová konstanta
  LMN_PER  := PQW272       //zápis akčního zásahu přímo na analogový výstup na adrese 272
```

Obr. 2.11 – Hlavní nastavení FB41

```
L     DB2.DBW    2         //načtení proměnné typu WORD z datového bloku 2 na adrese 2
DTR                        //převod proměnné na typ REAL
T     DB2.DBD    8         //proměnné typu REAL na adresu 8 v datovém bloku 2

L     DB2.DBD    8
L     5.714000e-001        //načtení reálné konstanty
*R                         //násobení dvou reálných čísel
T     DB2.DBD   20         //zápis výsledku na adresu 20 datového bloku 2

L     DB2.DBD   20
RND                        //přvod proměnné typu REAL na INT
T     DB2.DBW   30

L     DB2.DBW   30
T     "u2"                 //zápis proměnné na výstup analogového modulu
```

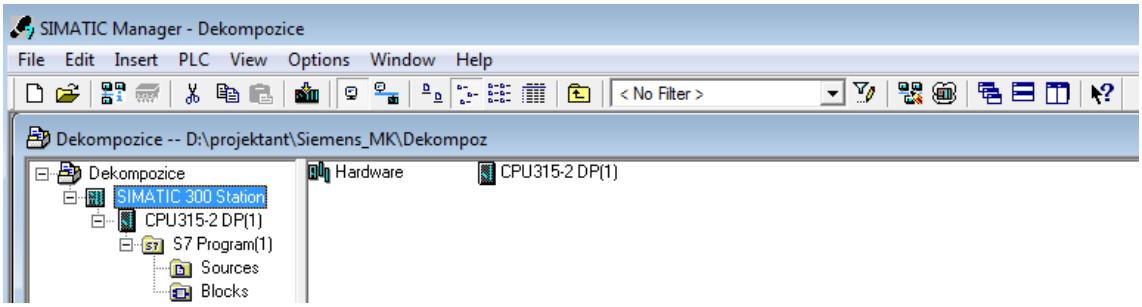
Obr. 2.12 – Příklad STL jazyka

Programování je pouze část realizace řízení, protože je potřeba ještě implementovat vytvořený program do PLC. Proto je velice důležité v programu Step 7 vytvořit hardwarovou konfiguraci, která odpovídá reálné sestavě PLC, a tuto konfiguraci následně zavést do PLC tak, aby konfigurace v PLC a Step 7 byly stejné. Tímto získám možnost sledovat proměnné v programu, které obsluhuje PLC nebo mohu sledovat stav PLC, pokud dojde k chybě. V hardwarové konfiguraci také mohu nastavit dobu vykonávání organizačních bloků *OB1* a *OB35*. V případě neprovedené hardwarové konfigurace program upozorní, že tato konfigurace není stejná a sledování proměnných, celkového stavu PLC a jeho programování by tak nebylo možné.

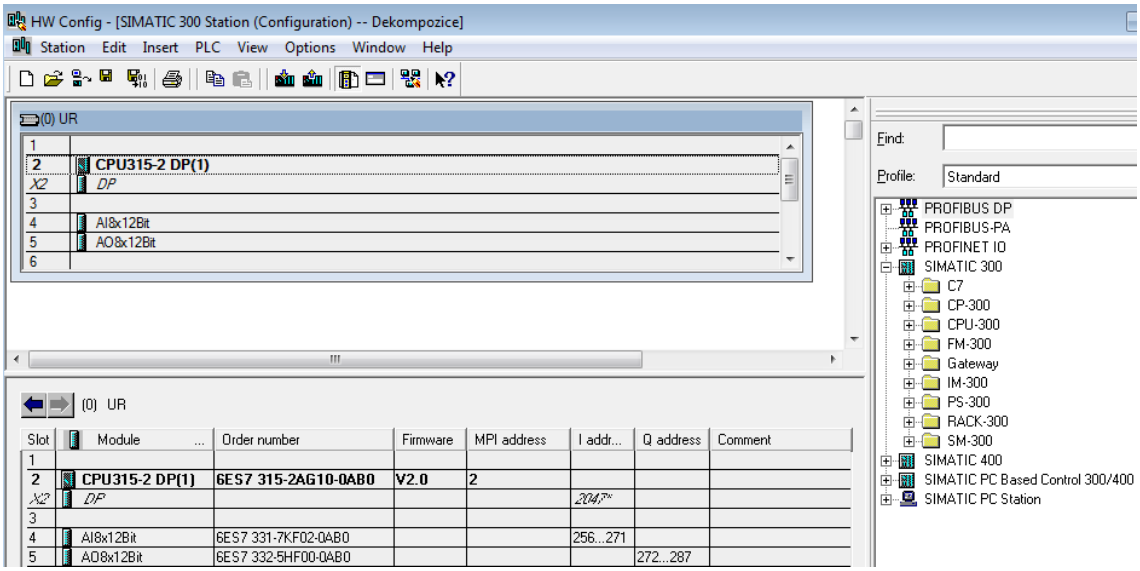
Vytvořený projekt se spravuje pomocí simatic manažeru (obr. 2.13), kde je vidět, z jakých částí se projekt skládá. V části, kde je definovaná řada PLC, respektive Simatic 300 Station, je nutné provést hardwarovou konfiguraci (obr. 2.14). V této části

je potřeba vybrat správné zařízení a moduly, které se nacházejí v pravé části, kde se vybírají většinou podle sériového čísla modulu, a následně se přesunou do lišty vlevo nahoře. Zde je možné vidět reálnou konfiguraci, kde jsem vybral použité CPU a analogové moduly. V dolní části lze ještě podrobněji zjistit, adresy vstupů a výstupu jednotlivých modulů, které lze samozřejmě i v této konfiguraci změnit. Po správném nastavení je potřeba konfiguraci uložit, zkompilovat a nahrát do PLC. Při konfiguraci reálného PLC jsem nemohl vybrat stejné sériové číslo použitého CPU, ale program mě na to upozornil, že sériové číslo není stejné, ale omezení při programování a obsluze PLC jsem nezaznamenal.

Dále se v simatic manažeru nachází samotný program (S7 Program), který se skládá ze zdrojových kódů, pokud se programuje v některém z vyšších programovacích jazyků, a z bloků, kde jsou obsaženy všechny bloky, které jsou pro realizování funkce PLC potřebné. Nacházejí se zde bloky organizační *OB1* a *OB35*, funkční blok PID regulátoru a několik datových bloků, případně *VAT* tabulka proměnných.



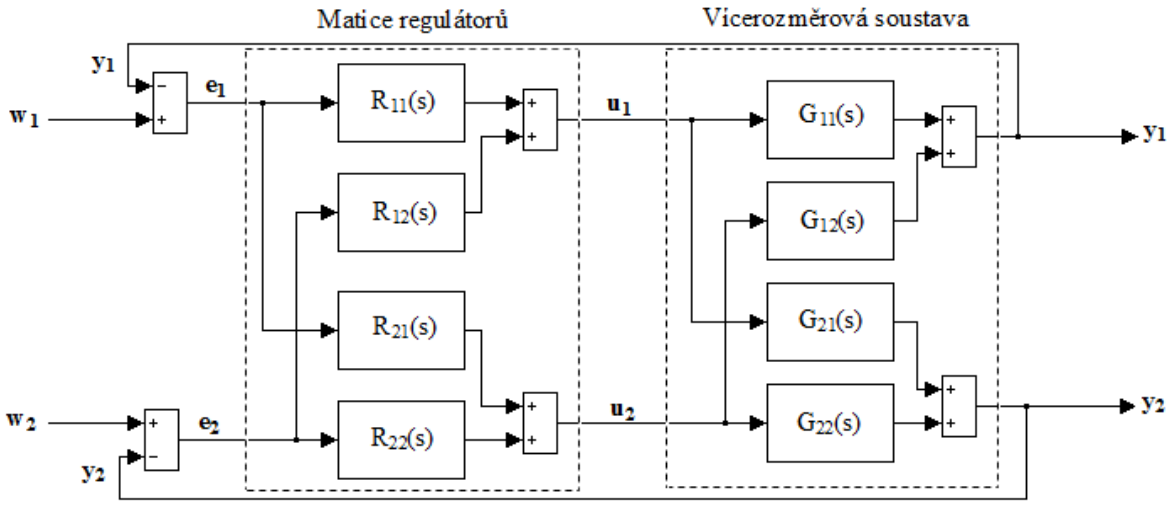
Obr. 2.13 – Struktura projektu



Obr. 2.14 – Hardwarová konfigurace

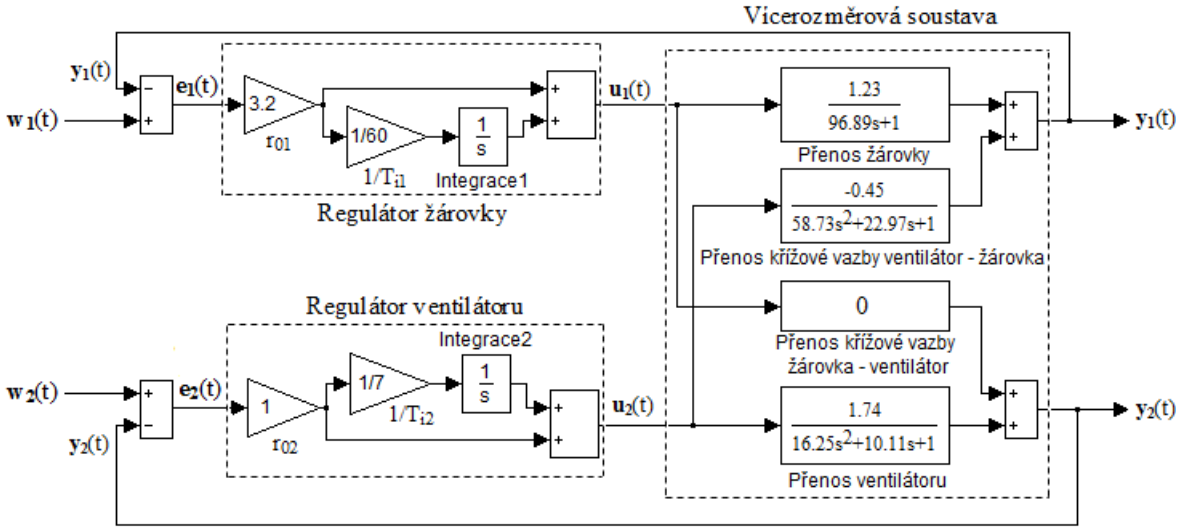
2.4 Decentralizované řízení

Řízení vícerozměrových soustav lze rozdělit na dva způsoby, kde jedna varianta je centralizované řízení a druhá možnost je decentralizované řízení. V případě, že se bude jednat o centralizované řízení, návrh regulátoru bude složitý a jeho implementace ještě více, protože celou vícerozměrovou soustavu řídí jeden regulátor, který je tvořen maticí regulátorů (obr. 2.15). Výhoda jediného regulátoru je v tom, že zahrnuje i přenosy regulátorů pro křížové vazby, a tak jejich správným nastavením se mohou potlačit křížové vazby mezi systémy.



Obr. 2.15 – Obecné schéma zpětnovazební smyčky s maticí regulátorů

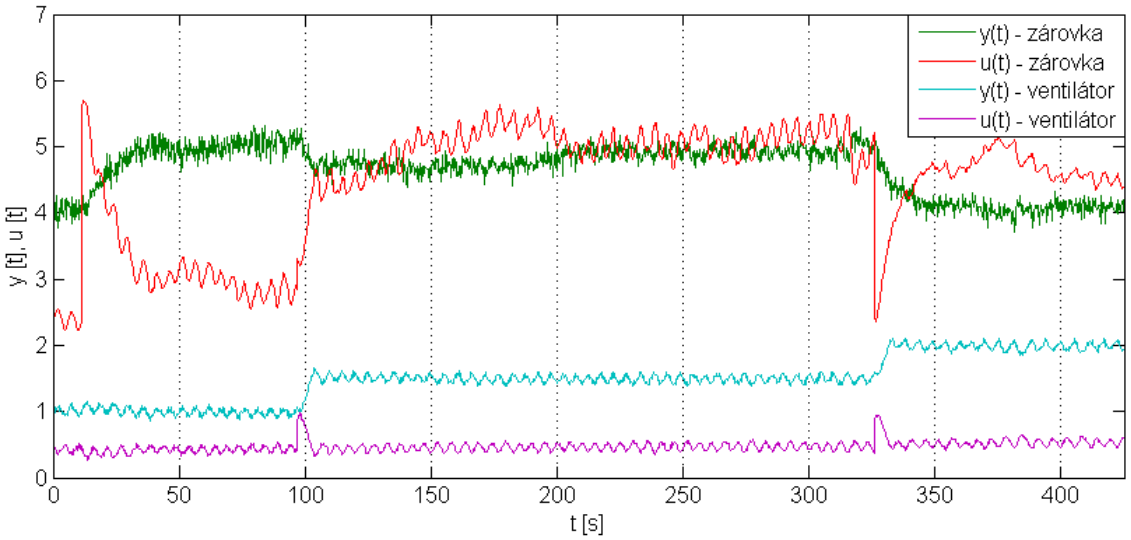
Princip decentralizovaného řízení spočívá v tom, že každý systém v soustavě je řízený vlastním regulátorem (obr. 2.16), jakoby se jednalo o SISO zpětnovazební regulační smyčky. Výhoda tohoto způsobu je podstatně jednodušší návrh regulátorů oproti centralizovanému řízení. Decentralizované řízení má ale i nevýhodu v tom, že pro potlačení křížových vazeb je potřeba upravit regulační strukturu, respektive rozšířit regulační obvod o určitou část, která bude realizovat eliminaci křížových vazeb. Tímto způsobem se zabývá dekompozice. Pro celkové řízení vícerozměrového systému je možnost decentralizovaného řízení jednodušší než centralizovaný způsob, proto se v praxi více využívá tato varianta. Při realizaci decentralizovaného řízení jsem vycházel ze struktury regulačního obvodu (obr. 2.16), kde jsem do PLC implementoval dva regulátory, které řídily dané systémy. Tímto způsobem řízení jsem ověřil reakce systémů, kde jsem sledoval změny regulovaných a akčních veličin jednotlivých samostatných systémů a také jsem naměřil závislost systémů mezi sebou, kde se projevila křížová vazba mezi ventilátorem a žárovkou (obr. 2.17).



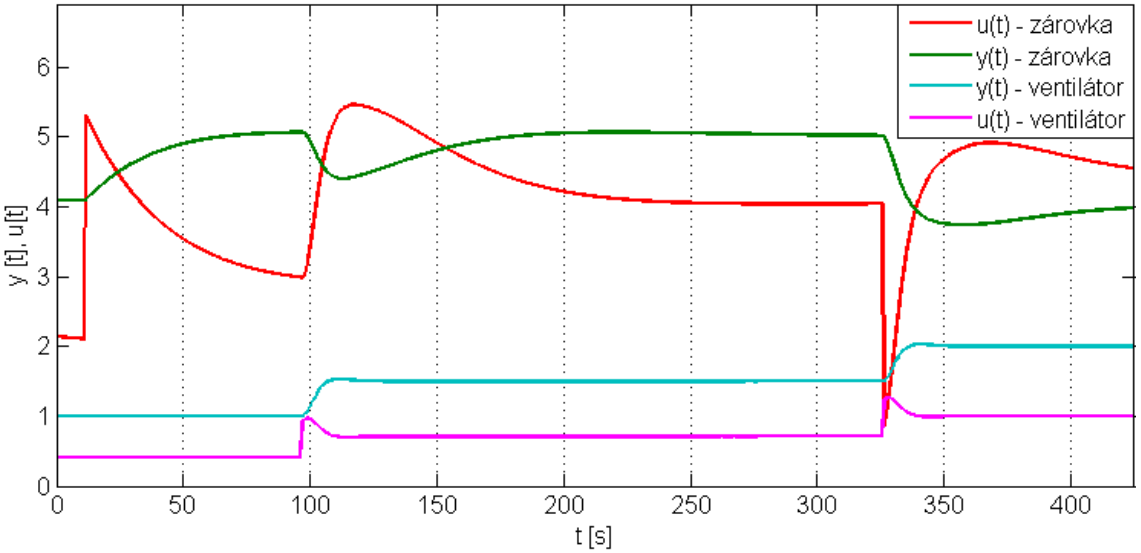
Obr. 2.16 – Obecné schéma regulační smyčky decentralizovaného řízení

Z naměřených reakcí jsem zjistil, že závislost teploty žárovky na otáčkách ventilátoru je silně nelineární, protože se mění zesílení přenosu křížové vazby v rozmezí 0,45–5, a také zesílení přenosu ventilátoru přibližně v rozmezí 1–4. Toto rozmezí jsem zjistil experimentálně při porovnávání naměřených (obr. 2.17) a simulovaných průběhů (obr. 2.18). Při měření decentralizovaného řízení jsem musel přistoupit na snížení otáček ventilátoru, protože při hodnotě pracovního bodu má ventilátor velmi vysoké otáčky a měření teploty na žárovce by nebylo adekvátní, protože rozsah dosažitelných teplot na žárovce by se výrazně snížil. Proto je ventilátor regulován na malé otáčky, aby se při měření teploty projevily změny na žárovce. Z důvodu nelineární závislosti otáček ventilátoru na teplotě se mohou v určitých částech simulované a naměřené průběhy odlišovat.

Měření decentralizovaného řízení (obr. 2.17) jsem provedl tak, že jsem nejdříve oba systémy nastavil na nenulové hodnoty, u ventilátoru na žádanou hodnotu 1 V a u žárovky na žádanou hodnotu 4 V. Poté jsem změnil žádanou hodnotu žárovky na 5 V a po ustálení regulované veličiny jsem zvýšil otáčky ventilátoru, kde lze vidět projev křížové vazby. Nakonec jsem provedl změnu žádaných hodnot u obou systémů, u žárovky na 4 V a u ventilátoru na 2 V.



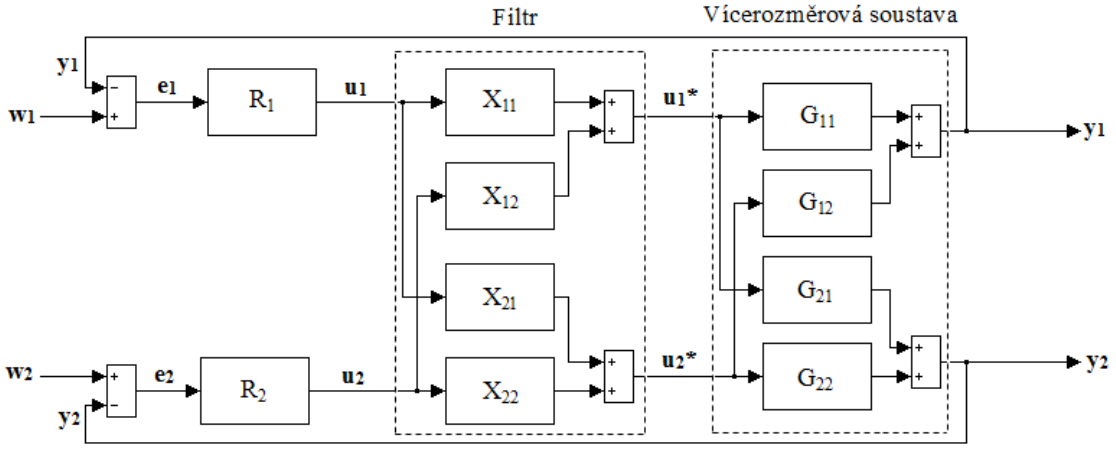
Obr. 2.17 – Naměřené průběhy decentralizovaného řízení



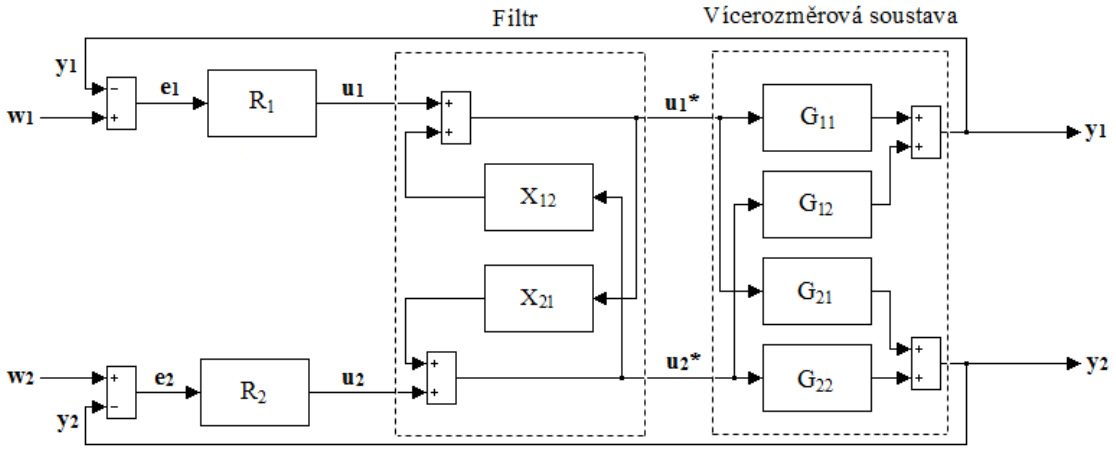
Obr. 2.18 – Simulované průběhy decentralizovaného řízení

2.5 Dekompozice

Dekompozice rozšiřuje možnosti decentralizovaného řízení, kde do regulační struktury se zavádí speciální filtr (obr. 2.19), který v ideálním případě zcela potlačí křížové vazby ve vícerozměrovém systému. Dekompozice může být realizována několika způsoby, podle kterých se rozlišuje výpočet filtru. Dekompozice může být ideální, statická, zjednodušená nebo inverzní. Z dostupných možností jsem vyzkoušel dekompozici inverzní a dekompozici statickou. Inverzní dekompozice (obr. 2.20) zcela potlačuje poruchové vlivy křížových vazeb, ale výpočet filtru je složitější než u statické dekompozice a následná implementace filtru do procesu reálného řízení je velmi obtížná.



Obr. 2.19 – Obecné schéma dekompozice



Obr. 2.20 – Obecné schéma inverzní dekompozice

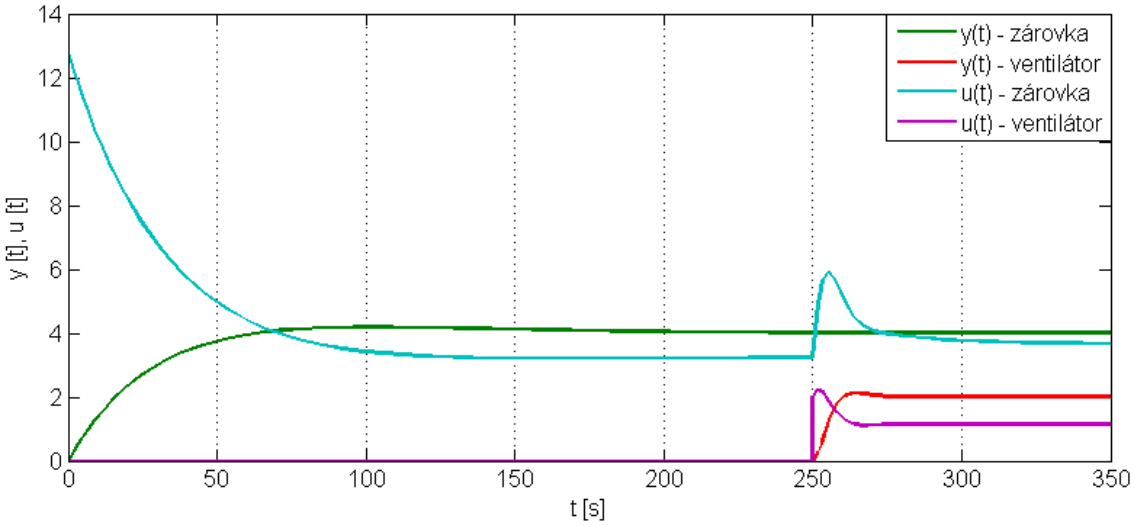
Výpočet filtru pro inverzní dekompozici se provádí podle definovaných vztahů (2.4) a (2.5). Výhodou reálné vícerozměrové soustavy je, že jedna křížová vazba je nulová, proto se v simulaci uvažuje pouze jeden filtr, a to filtr X_{12}, protože filtr X_{21} při pohledu na definovaný vztah zřetelně vychází nulový, protože přenos G_{21} je nulový.

$$X_{12} = -\frac{G_{12}}{G_{11}} \tag{2.4}$$

$$X_{21} = -\frac{G_{21}}{G_{22}} \tag{2.5}$$

$$X_{12} = \frac{43{,}6s+0{,}45}{72{,}24s^2+28{,}25s+1{,}23} \tag{2.6}$$

Po dosazení příslušných přenosů do definovaných vztahů pro výpočet filtru inverzní dekompozice jsem vypočítal filtr 2. řádu (2.6). Filtr, který je dán přenosem, je velmi obtížné implementovat do PLC, proto jsem provedl pouze simulaci inverzní dekompozice (obr. 2.21). Ze simulace je na první pohled zřejmé, že inverzní dekompozice zcela potlačila rušivé vlivy způsobené ventilátorem, protože v čase, kdy se ventilátor spustí, nedochází k ovlivnění regulované veličiny žárovky.



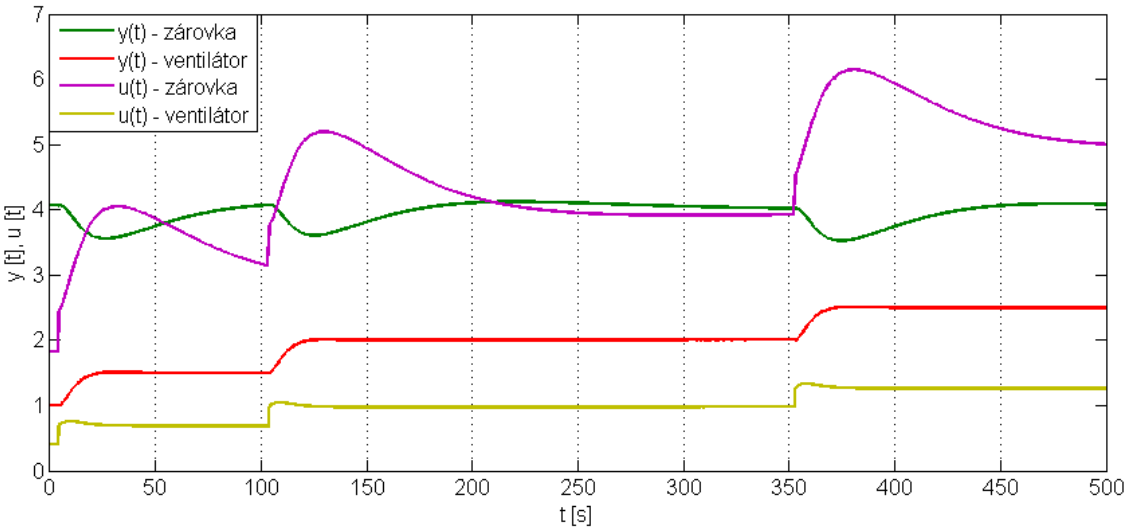
Obr. 2.21 – Simulace inverzní dekompozice

Pro možnost aplikace dekompozice na reálnou soustavu jsem zvolil statickou dekompozici, která svojí strukturou odpovídá obecnému filtru (obr. 2.19), kde filtr je tvořen maticí (2.7), ale hodnoty jednotlivých prvků v matici jsou dány pouze konstantním zesílením. Omezení u statické dekompozice je takové, že musí k matici přenosů soustavy existovat matice inverzní, protože při výpočtu matice filtru je potřeba znát inverzní matici přenosů a druhá matice, která se uplatňuje při výpočtu je jednotková matice, která má nenulové prvky pouze na diagonále. Dalším omezením statické dekompozice je, že systém by měl být v pracovní oblasti lineární, jinak se reálné výsledky nebudou s velkou přesností shodovat se simulacemi. Jednotkovou matici si mohu zvolit, kde nejvhodnější způsob volby prvků na diagonále je zvolit prvky rovné jedné.

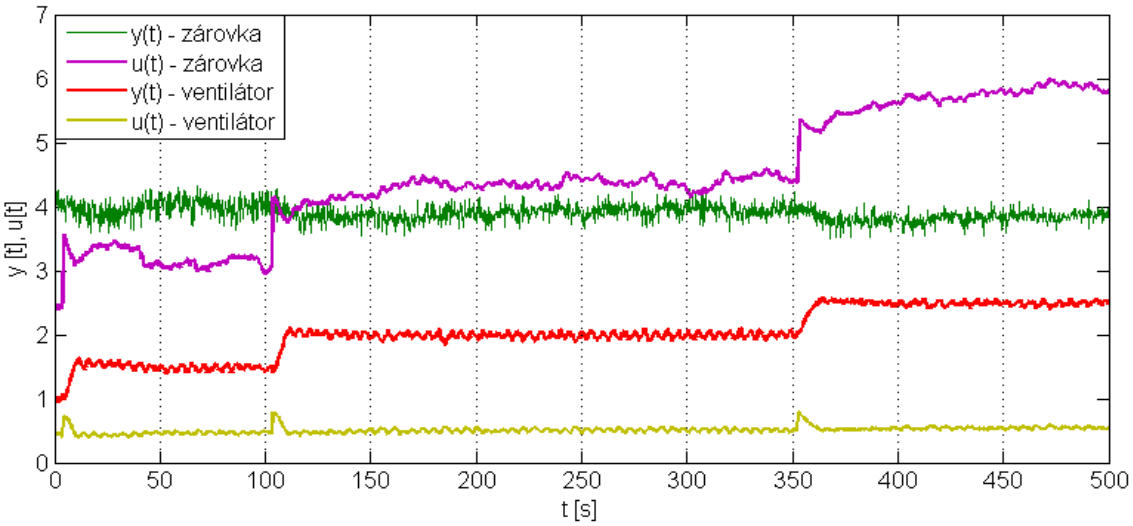
$$X(0) = G(0)^{-1} \cdot \begin{bmatrix} S_{11}(0) & 0 \\ 0 & S_{22}(0) \end{bmatrix} \tag{2.7}$$

$$X(0) = \begin{bmatrix} 1{,}23 & -0{,}45 \\ 0 & 1{,}74 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}81 & 0{,}21 \\ 0 & 0{,}57 \end{bmatrix} \tag{2.8}$$

Z provedeného výpočtu (2.8) je zřejmé, že statická dekompozice je skutečně realizována pouze maticí filtrů s konstantními prvky, a proto jsem tuto variantu vyzkoušel při řízení vícerozměrové reálné soustavy. Statická dekompozice v principu neslouží k úplnému potlačení poruchových vlivů křížových vazeb, ale většinou částečně omezuje vliv křížových vazeb, čímž vytváří možnost navrhnout regulátory pro decentralizované řízení. Protože jsem zvolil jednotkovou matici s prvky na diagonále rovné 1, využil jsem při řízení dříve navržené regulátoru pro decentralizované řízení. Statickou dekompozici jsem implementoval do PLC a následně jsem naměřil reakce systému žárovky na vliv poruchy od ventilátoru (obr. 2.23), které jsem poté porovnal se simulací (obr. 2.22). Při měření statické dekompozice a následném porovnání průběhu se stále projevoval vliv nelinearity způsobené závislostí otáček ventilátoru na teplotě. Proto při srovnání průběhů se simulacemi docházelo v některých částech měření k rozdílům.

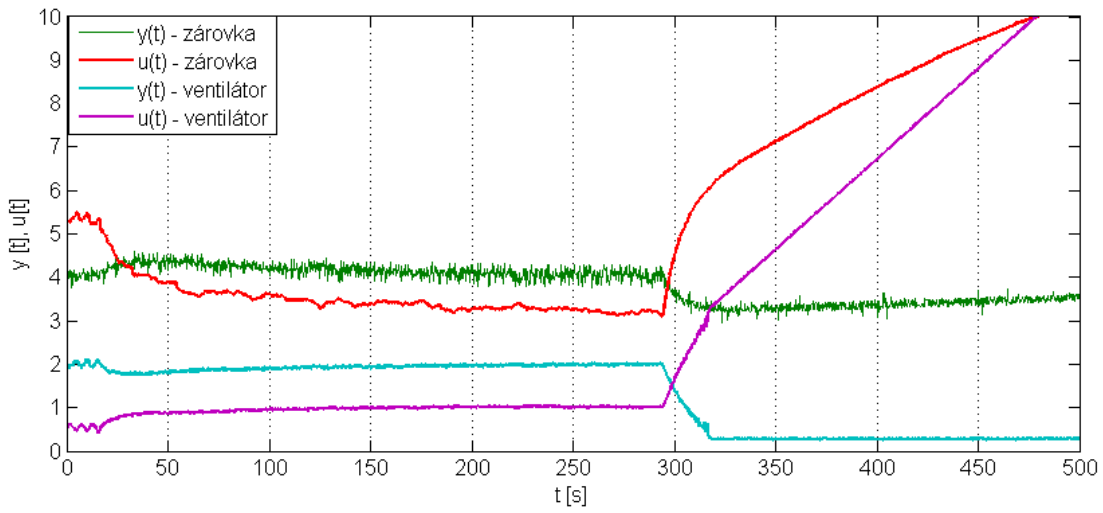


Obr. 2.22 – Simulace statické dekompozice



Obr. 2.23 – Naměřené průběhy statické dekompozice

U dekompozice jsem ještě vyzkoušel, jakou reakci bude mít soustava na chybu ventilátoru způsobenou zakrytím přívodu vzduchu (obr. 2.24). Měření jsem provedl tak, že jsem nejdříve nastavil nenulové hodnoty žádaných veličin systémů, a po ustálení jsem částečně zakryl přívod vzduchu ventilátoru, kde z naměřených průběhů lze poznat pokles průtoku vzduchu, který se po určité době vrátí zpět na žádanou hodnotu. Následně jsem zakryl přívod vzduchu celkově, takže ventilátor neměl žádný přísun vzduchu, a z naměřených dat je vidět, že průtok vzduchu klesl na nulovou hodnotu a regulátor se snaží zvyšovat akční zásahy, aby regulovaná veličina zůstala na žádané hodnotě, ale ventilátor nemůže plnit svoji funkci, protože nemá přívod vzduchu. Žárovka je chybou ovlivněna také, ale není tolik výrazná než při standardním průtoku vzduchu, nicméně vlivem dekompozice dochází ke zvyšování akčních zásahů pro žárovku současně.



Obr. 2.24 – Měření chyby na ventilátoru

3 Komunikace s PLC

V situacích kdy se nemusí programovat funkce PLC, ale vzniká potřeba sledování stavu PLC, jeho nastavování nebo případné sledování různých proměnných, tak se využívá programů, jejichž obsluha je v některých případech velmi jednoduchá. V současnosti se pro komunikaci s průmyslovými zařízeními využívá komunikační protokol OPC, který dále komunikuje s OPC klientem.

Pokud bych se snažil vytvořit vlastní komunikaci mezi PLC a počítačem na základě některého z programovacích jazyků, musel bych realizovat přenos dat v rámci komunikačního rozhraní, které využívá PLC, respektive bych musel realizovat vše, co má OPC protokol už implementované. Proto jsem možnost vlastní komunikace zavrhnul a soustředil jsem se na využití OPC protokolu.

3.1 OPC server

OPC server slouží jako prostředník mezi komunikací průmyslového zařízení a OPC klientem v PC. Velkou výhodou OPC protokolu je, že zahrnuje velké množství komunikačních protokolů různých průmyslových zařízení, proto pomocí OPC serveru se může komunikovat s PLC automaty od různých výrobců, bez ohledu na jejich komunikační rozhraní. Následně lze využít několika vizualizačních aplikací, které umožňují komunikovat s OPC serverem, protože využívají ke komunikaci stejný protokol, kde OPC server tak může přenášet sledované proměnné z PLC do vizualizačního prostředí. OPC server mimo sledování proměnných vytváří možnost zápisu nových hodnot do PLC.

V době kdy nebyla možnost využívat OPC protokol, byla nutnost pro každé zařízení instalovat zvlášť ovladače, kde při větším počtu zařízení, se kterými bylo potřeba komunikovat, mohlo docházet k různým problémům s kompatibilitou ovladačů. OPC protokol tyto problémy odstraňuje, protože jediné komunikační rozhraní je realizováno pomocí OPC. Pokud chci využívat OPC protokol, je potřebné mít nainstalované 2 programy v počítači, kde první program je OPC server, který realizuje komunikaci mezi zařízením a počítačem, kde se zařízením komunikuje jeho protokolem a získaná data převádí do formy OPC. Druhý program je OPC klient, který přijímá data ve formě OPC ze serveru a následně může provádět jejich vizualizaci nebo záznam.

V současnosti je k dispozici velké množství OPC serverů od různých výrobců od profesionálních řešení, které jsou zpoplatněny vysokými částkami až po servery, které jsou zdarma k dispozici a pro běžné účely dostačující, ale mají omezenou dobu, po kterou může být server spuštěn. Po uplynutí této doby musí být OPC server restartován. Mezi nejznámější OPC servery patří např. INAT (Industrial Networks for Automation Technology), dále servery od firmy Kepware nebo Deltalogic a samozřejmě Siemens poskytuje vlastní OPC server WinCC. Protože jsem našel jednoduché řešení OPC serveru od firmy Deltalogic, kde server má sice omezenou dobu spuštění na 90 minut, ale při použití OPC klienta od firmy Reliance jsem nepoznal toto časové omezení, tak jsem po vyzkoušení daného serveru zjistil, že je dostačující a další varianty jsem nemusel využít.

Instalace a nastavení OPC serveru od Deltalogic je velice jednoduchá a rychlá. Velkou výhodou serveru je, že při konfiguraci se zavádí cesta k vytvořenému projektu v programu Step 7, takže všechny proměnné a datové bloky, které v projektu používám, mohu jednoduše přes OPC server sledovat. Podrobnější postup nastavení viz příloha 2. Po úspěšném nastavení OPC serveru jsem mohl přistoupit k vytvoření vizualizace pomocí programu Reliance.

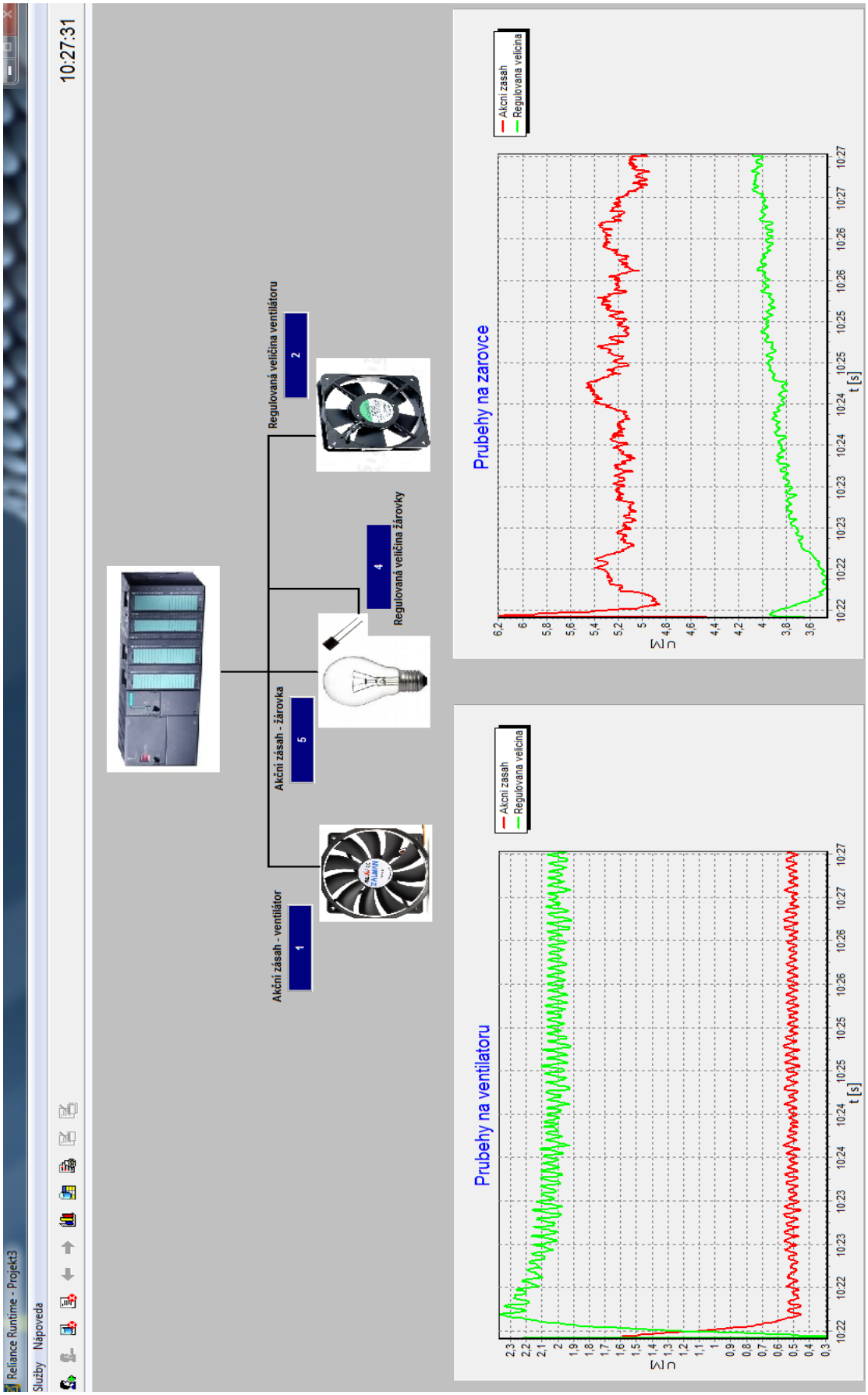
.

3.2 Vizualizace a záznam dat

Pro vizualizaci a záznam dat jsem využil OPC klienta, který vytvořila česká firma Reliance, kde tato firma mimo jiné vytváří i vlastní OPC servery. OPC klient je v několika verzích, kde nejnovější verze je Reliance 4, ale já jsem použil o verzi starší aplikaci, a to Reliance 3 ve verzi 3.7.7. OPC klient je standardně nabízen pro komerční využití zdarma, ale je omezen na 25 datových bodů, které lze sledovat, ale tento počet bodů mě při realizaci sledování dat neomezoval, protože jsem využíval pouze 4 body. Pokud by uživatel potřeboval více datových bodů pro sledování, je to možné, ale tato možnost je zpoplatněna, kde maximální počet sledovaných bodů je uváděn nad 10 000, ale ve většině případů je zcela dostačující počet 500 datových bodů.

Výhodou OPC klienta je možnost komunikace s použitým OPC serverem firmy Deltalogic, proto je nastavení sledování veličin velmi jednoduché. OPC klient Reliance se skládá z několika podprogramů, které slouží pro návrh vizualizace, pro spuštění vizualizace atd. Nejčastěji jsem využíval program Reliance designe, kde se provádí návrh vizualizačního prostředí a druhý program, který jsem nejčastěji používal, byl Reliance runtime, který spouští vytvořenou vizualizaci. Program Reliance designe umožňuje spuštění Reliance runtime přímo ze svého prostředí, ale zde jsem narazil na problém, protože docházelo k jisté chybě v knihovně, protože program při pokusu o spuštění hlásil, že nemůže najít součást knihovny. Proto jsem spuštění vizualizace vyřešil tak, že jsem uzavřel program Reliance designe, a spustil jsem ručně program Reliance runtime, kde jsem vybral umístění projektu vizualizace a poté jsem úspěšně spustil vytvořenou vizualizaci.

Provázání programu Reliance s OPC serverem se provádí několika kroky, kde nejdříve se ve správci stanic připojí k nově vytvořenému projektu nakonfigurovaný OPC server, kde se také vybírají všechny proměnné, které chci sledovat. Poté se vytvoří databáze, do které se budou ve vizualizaci ukládat sledovaná data a později je možné tyto data přenést do textového souboru. Tento postup je základní proces při navázání komunikace OPC klienta a serveru. Program Reliance obsahuje množství variant, jakými lze sledované veličiny zobrazit, využil jsem zobrazení ve formě plovoucího grafu, který v reálném čase zobrazoval průběhy příslušných veličin a ve formě textové, kde se na displeji zobrazovaly aktuální hodnoty sledovaných veličin. Podrobnější postup nastavení jsem uvedl do přílohy 3. Výsledná vizualizace, kterou jsem realizoval (obr. 3.1), ukazuje ve zjednodušené formě zapojení soustavy a sledované veličiny.



Obr. 3.1 - Realizovaná vizualizace

Závěr

Výsledkem mojí bakalářské práce je splnění všech bodů zadání. Nejdříve jsem prováděl identifikaci soustavy, kde se zpočátku objevily problémy při měření statických charakteristik, kterými byl ovlivněn především systém žárovky. Ze senzorů teploty jsem získával znehodnocená data, protože v začátku měření jsem získal odezvy, které se velmi dlouho ustalovaly, a také zde docházelo ke vzniku zákmitů. Systém žárovky měl takové chování, že při nulových otáčkách ventilátoru senzory teploty nedosáhly maximálního rozsahu 10 V, pro některé senzory byla maximální hodnota kolem 6 V. Tento problém jsem konzultoval s vedoucím bakalářské práce a po odstranění problému jsem provedl nová měření pro identifikaci. Po získání přenosů jsem ve fázi řízení zjistil, že některé simulace v porovnání s reálnými systémy se odlišují, proto jsem byl nucen provést zpětně nové měření pro získání přenosů ventilátoru a křížové vazby. Identifikací nových měření jsem dosáhnul požadovaných přenosů, které v porovnání s reálnými systémy jsou přijatelné.

V další části jsem úspěšně navrhnul a aplikoval PI regulátory do použitého PLC pro decentralizované řízení, kde jsem poté naměřil odezvy reálných systémů a jejich vzájemné závislosti. Pro programování PLC jsem nejdříve zkoušel vyšší programovací jazyk SCL, který jsem podle způsobu zápisu instrukcí považoval za nejlepší volbu, ale zjistil jsem, že realizace řízení pomocí SCL jazyka není tak snadná. Snažil jsem se realizovat vlastní regulátor pomocí SCL kódu, ale narazil jsem na problém s přetypováním proměnných, kde překladač sice nehlásil žádné problémy, ale po aplikaci algoritmu do PLC se mi nepodařilo zprovoznit daný program. Proto jsem přistoupil k programování pomocí STL, který není zcela přehledný, ale vytvořený program zde funguje bez problémů. Protože jsem měl možnost využít implementovaného funkčního bloku PID regulátoru, pro decentralizované řízení jsem si vystačil pouze s tímto funkčním blokem a ve fázi realizace dekompozice jsem využíval pouze několik instrukcí v STL kódu. Úspěšně jsem se pokusil aplikovat možnost dekompozice, kde jsem si pro řízení vybral statickou dekompozici, kterou jsem také implementoval do PLC a naměřil příslušné odezvy.

V poslední části jsem využil komerčně dostupných programů OPC serveru a OPC klienta pro realizaci sledování a přenosu dat z PLC do PC. Protokol, který používají dané programy, je velice výhodný v tom, že nebylo potřeba rozšiřovat

hardware PLC, protože komunikační rozhraní PLC použitý OPC server podporuje. Úspěšně jsem se tak pokusil realizovat tento bod zadání.

Seznam použité literatury

[1] BERGER, Hans. *Automating with STEP 7 in STL and SCL: programmable controllers SIMATIC S7-300/400.* Germany, 2005.

[2] BLAŽEK, Jaroslav. *Kurz programování PLC SIEMENS SIMATIC S7–300.* [online]. [cit. 15. května 2012]. URL: <http://www.foxon.cz/index.php?main_page=faq_info&fcPath=30_31&faqs_id=167ca>.

[3] BLAŽEK, Jaroslav. *S7 OPC Server od Deltalogic pro PLC Simatic S7–200/300/400.* [online]. [cit. 15. května 2012]. URL: <http://www.blaja.cz/index.php?option=com_content&task=view&id=151&Itemid=52>.

[4] DRŽKOVÁ, Markéta. *První kroky*. [online]. [cit. 15. května 2012]. URL: <http://www.reliance.cz/files-to-download/documentation/reliance3/Reliance3_FirstSteps_CSY.pdf>.

[5] DRŽKOVÁ, Markéta. *Reliance design*. [online]. [cit. 15. května 2012]. URL: <http://www.reliance.cz/files-to-download/documentation/reliance3/Reliance3_Design_CSY.pdf>.

[6] FOXON s.r.o. *OPC servery*. [online]. [cit. 15. května 2012]. URL: <http://www.foxon.cz/opc-server-opc-klient-c-72.html>.

[7] HLAVA, Jaroslav. *Skripta PAR*. [online]. [cit. 15. května 2012]. URL: <http://www.fm.tul.cz/~jaroslav.hlava/par/Skripta_PAR.pdf>.

[8] HUBKA, Lukáš. *Real Time Toolbox*. [online]. [cit. 15. května 2012]. URL: <http://www.fm.tul.cz/esf0247/download_file.php/Real_Time_Toolbox.pdf?id=76>.

[9] HUBKA, Lukáš. *Řízení vícerozměrových systémů: Dekompozice.* Liberec, 2012.

[10] KOCHANICEK. *Nastavení hardwarové konfigurace pro CPU 314C-2DP v programu SIMATIC Manager*. [online]. [cit. 15. května 2012]. URL: <http://coptel.coptkm.cz/reposit.php?action=0&id=32370&revision=-1&instance=2>.

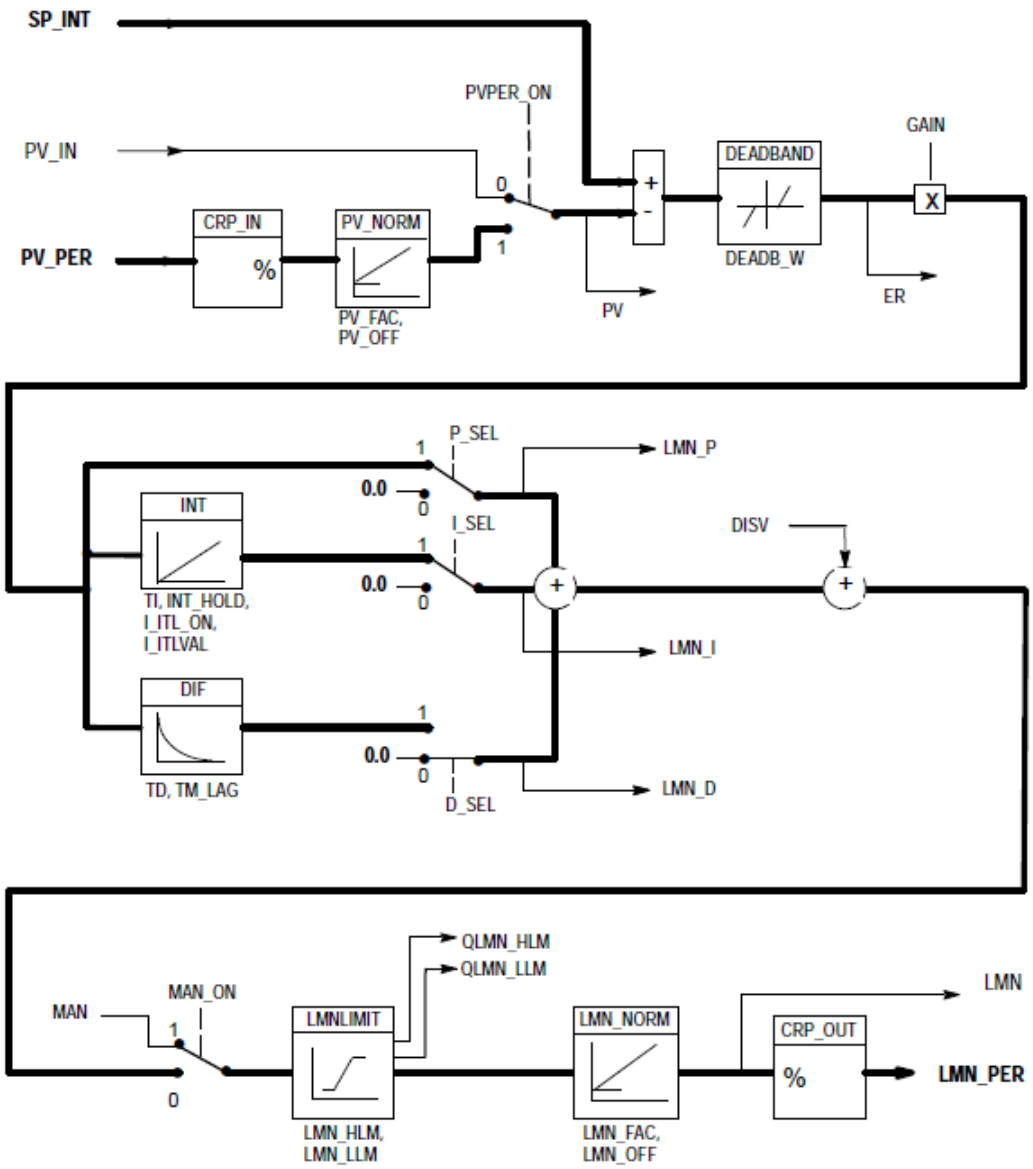
[11] MODRLÁK, Osvald a Lukáš HUBKA. *Řízení vícerozměrového systému.* Liberec, 2008.

[12] SIEMENS. *Simatic S7-300 SM331; AI 8x12 Bit Getting Started*. [online]. [cit. 15. května 2012]. URL: <http://cache.automation.siemens.com/dnl_iis/DE/DEzMDU2NQAA_17473828_HB/s7300_sm331_ai_8x12_bit_getting_started_en-US_en-US.pdf>.

[13] SIEMENS. *Standard Software for S7-300 and S7-400 PID Control*. [online]. [cit. 15. května. 2012]. URL: < http://www.fer.unizg.hr/_download/repository/S7pidcob.pdf >

[14] VANEKOVÁ, K. *PID regulátory průmyslového řídicího systému SIMATIC*. [online]. [cit. 15. května 2012]. URL: <http://www.kirp.chtf.stuba.sk/~vanekova/prilohy/Funkcne_bloky_PID_regulatorov.pdf>

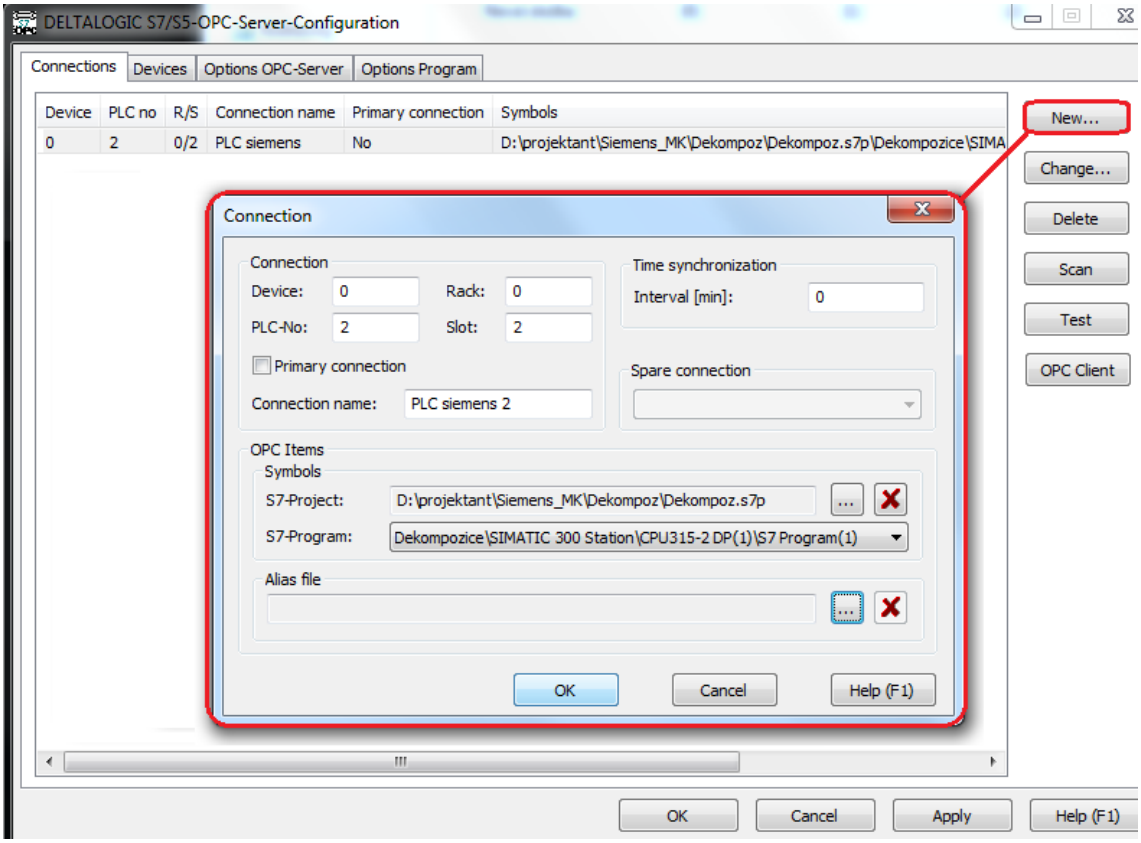
Příloha 1 – PID blok ve Step 7



Příloha 1.1 – Struktura funkčního bloku FB41 [13]

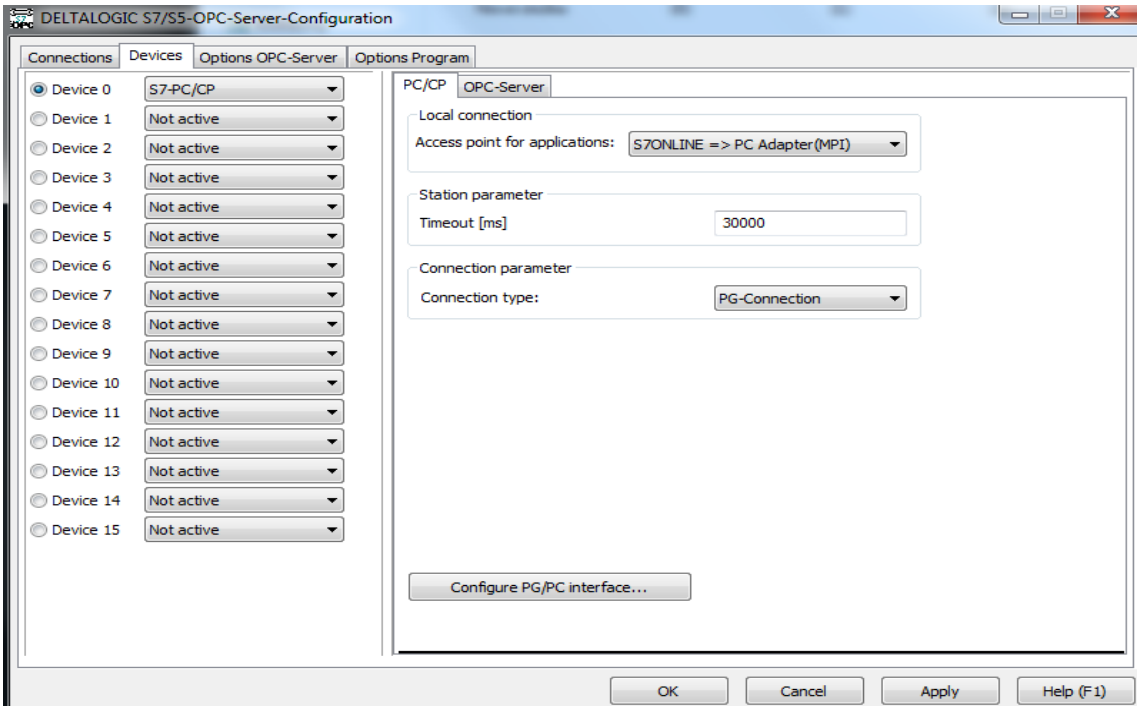
Příloha 2 – Nastavení OPC serveru

Pro správnou funkci OPC serveru je nutné provést jeho konfiguraci, která se spustí ikonou *Deltalogic S7-S5-OPC-Server-Configuration*. Po otevření konfigurace se v záložce *Connections* vytvoří nové spojení s PLC tlačítkem *New*, kde se nastaví všechny parametry PLC, jako adresa PLC, kterou udává adresa komunikačního rozhraní MPI v hardwarové konfiguraci, kde v tomto případě byla 2, nebo číslo zařízení, v jakém se nachází slotu atd. Dále se v *OPC Items* nastaví cesta, kde je uložený projekt z programu Step 7 a vše se potvrdí. V záložce *Connections* se zobrazí nastavené PLC a případně tlačítkem *Test* je možné provést kontrolu, jestli je spojení skutečně navázané.



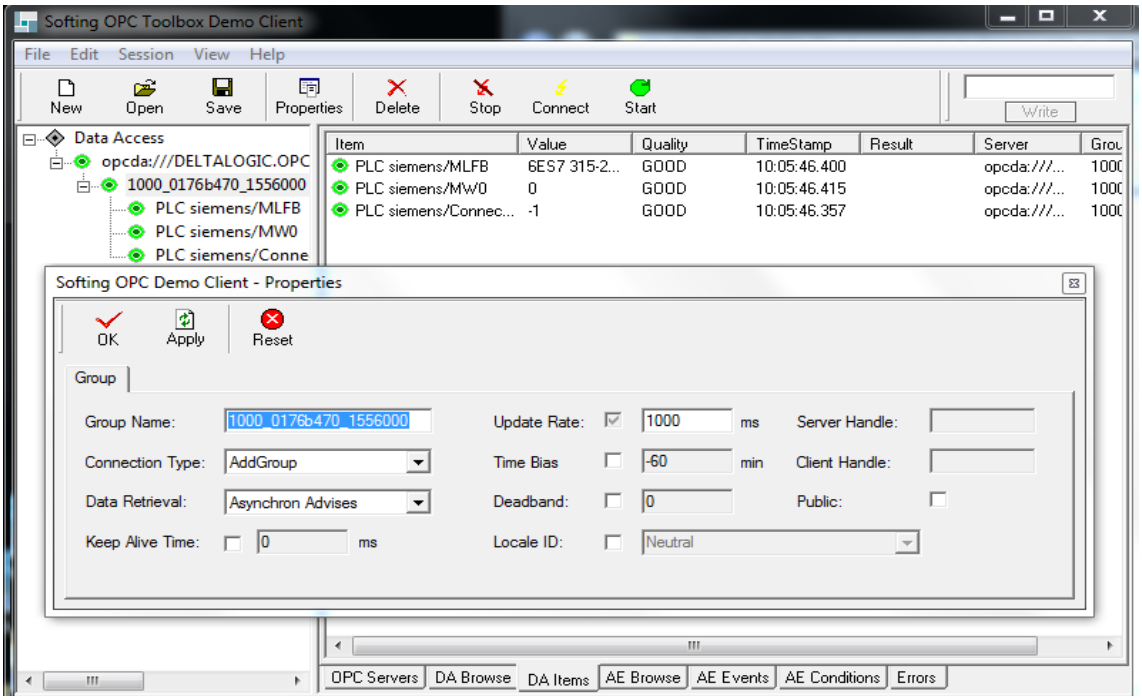
Příloha 2.1 – OPC server

Poslední nastavení, které je potřeba u serveru provést, je na záložce *Devices*, kde se nastaví typ komunikace mezi PC a PLC. Protože ke komunikaci využívám převodník MPI na USB, který se reprezentuje jako komunikační rozhraní *PG/PC*, proto jsem zvolil komunikaci *S7-PC/CP*. Po výběru správné komunikace je potřeba ještě nastavit doplňující parametry, jako je typ komunikačního rozhraní PLC, zde je nastaveno na *PC Adapter (MPI)* a také typ spojení, respektive *PG-Connection*.



Příloha 2.2 – Nastavení komunikace

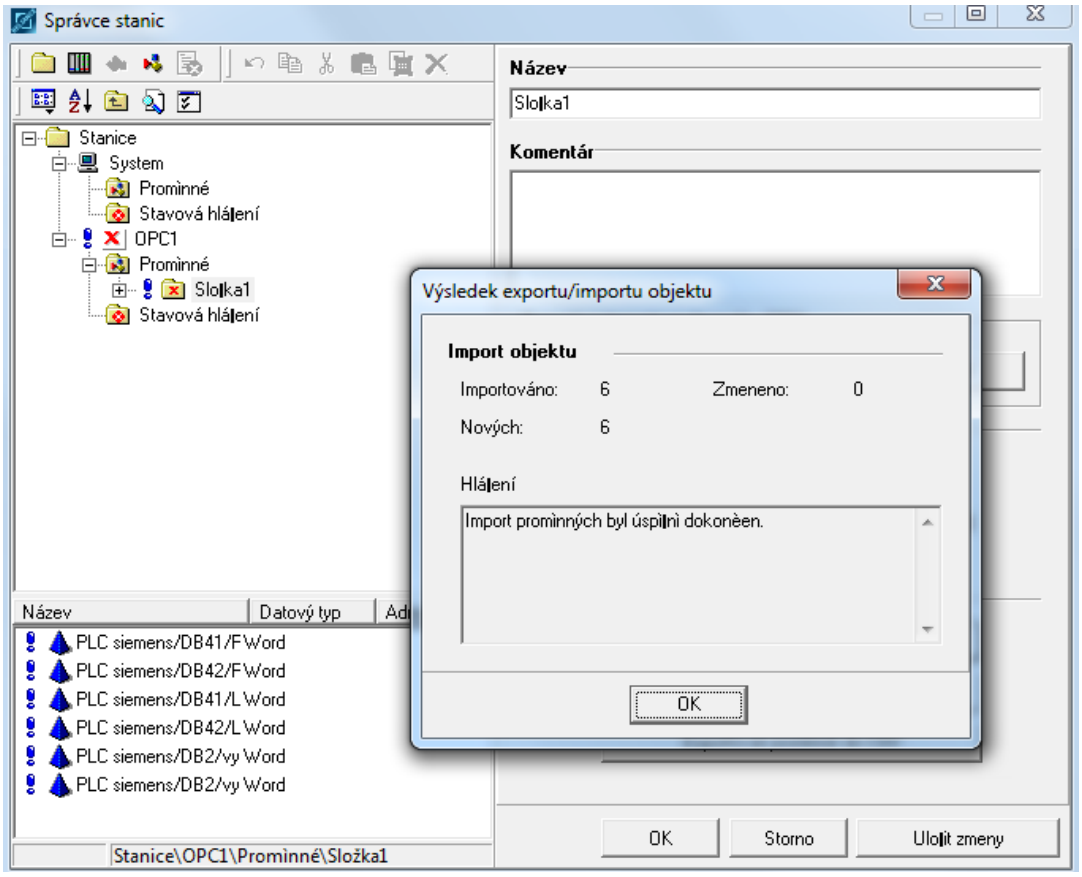
Po provedeném nastavení OPC serveru je možné spustit OPC klienta, který se spouští na záložce *Connections*, ale zde se nejedná o OPC klienta, který jsem využíval k vizualizaci, je to pouze klient, kde se může nastavit např. perioda odběru sledovaných dat.



Příloha 2.3 – Nastavení periody odběru dat

Příloha 3 – Nastavení OPC klienta

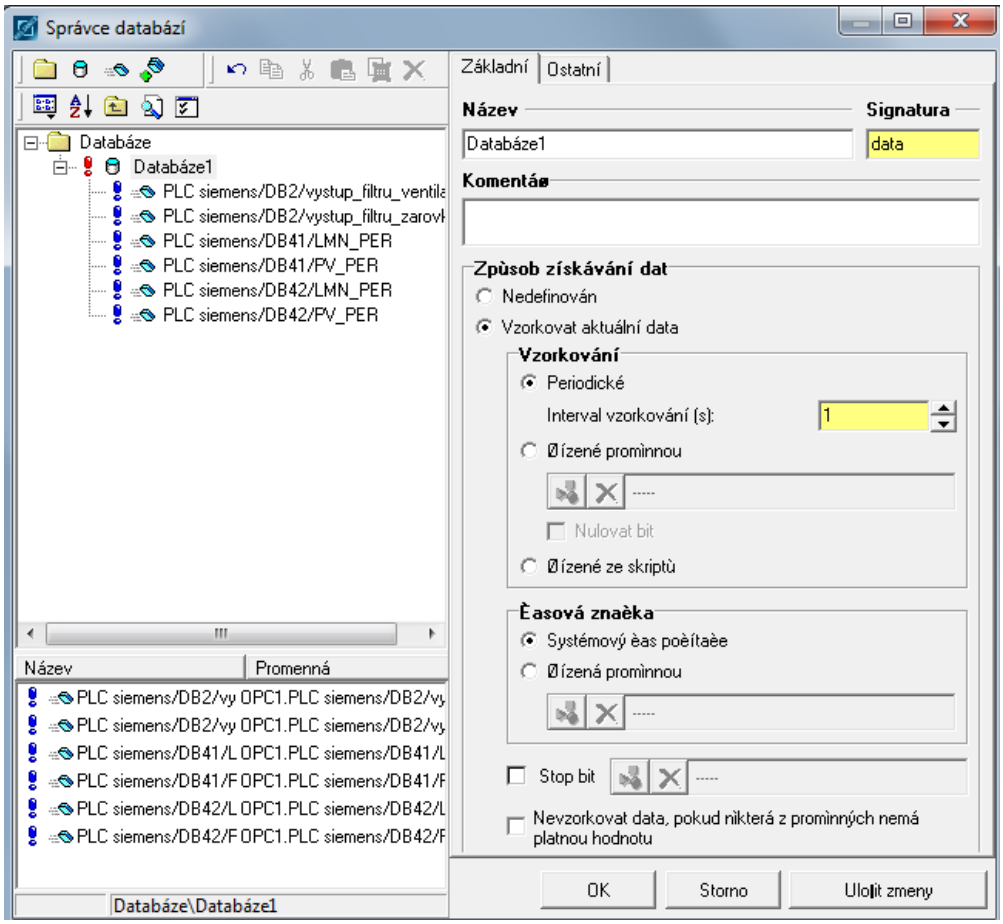
Všechna nastavení pro OPC klienta se provádějí pod záložkou *Správci*. Pro přiřazení OPC serveru k příslušnému klientovi se musí pod záložkou *Správci* otevřít *Správce stanic* (příloha 3.1). Nejdříve se v tomto správci klikne pravým tlačítkem myši na *Stanice* → *Nová stanice* → *OPC*, kde po vložení OPC je nutné ještě přiřadit daný OPC server. Provede se to tak, že po označení OPC se v pravé části, kde jsou Parametry, otevře nabídka pomocí tlačítka „…“, kde jsou k výběru OPC servery a mimo jiné se zde bude nacházet dříve nakonfigurovaný server Deltalogic. Poté se provede výběr sledovaných veličin, kde nejdříve je nutné pravým tlačítkem myši vybrat *Proměnné* (v OPC) → *Nová složka proměnných*. Po označení nově vytvořené složky a výběru tlačítka *Importovat proměnné z OPC serveru* v pravé dolní časti, se otevře okno, kde je možné vybrat v OPC serveru z přiřazeného projektu dané proměnné z datových bloků, které se jsou potřeba sledovat.



Příloha 3.1 – Správce stanic

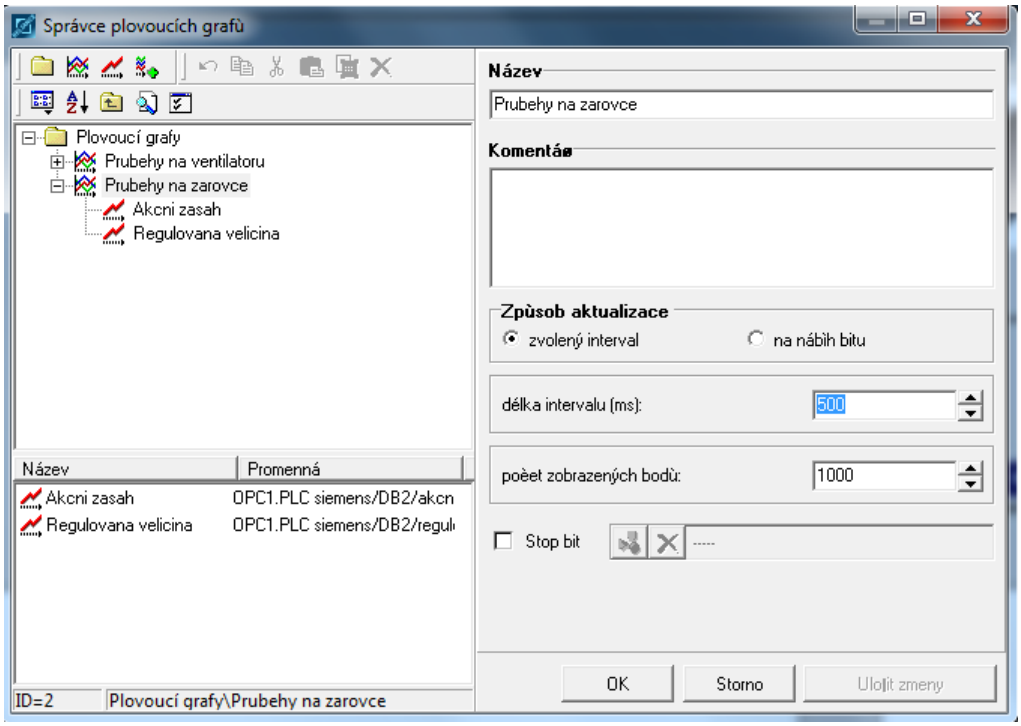
Další správce, který je potřeba nastavit je *Správce databází*, který slouží pro ukládání sledovaných proměnných a později umožňuje export dat. Po otevření je

nejdříve nutné vytvořit novou databázi tak, že se pravým klikne na *Databáze* → *Nová databáze*. Poté se do databáze přiřadí sledované veličiny tak, že se klikne pravým na vytvořenou databázi → *Přidat databázové položky*, kde se vyberou proměnné, které chceme sledovat, ale pouze ty, které jsme dříve přiřadily pomocí Správce stanice v OPC. Pro databázi se ještě nastaví tzv. *Signatura*, která udává pouze pojmenování databáze na pevném disku PC při ukládání, a dále se ještě nastavuje Vzorkování, které udává periodu ukládání sledovaných dat.



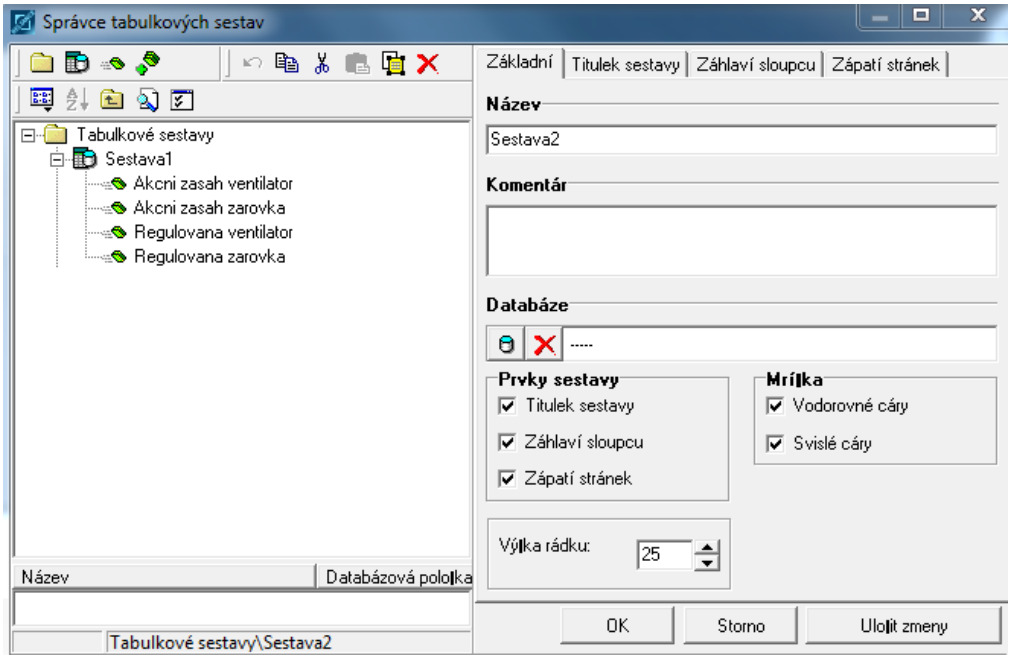
Příloha 3.2 – Správce databází

Pro přidání plovoucího grafu, který realizuje sledování veličin v reálném čase je potřeba přidat tento graf ve *Správci plovoucích grafů*. Zde se pravým tlačítkem vybere, po otevření Správce, *Plovoucí grafy* → *Nový graf*, kde následně se pravým tlačítkem vybere vytvořený graf → *Přidat řady grafu*. Následně se objeví nové okno, ve kterém se vybere umístění sledované proměnné, v tomto případě z OPC serveru, přičemž graf může mít přiřazené více než jednu proměnnou. Dále se ještě u grafu nastaví perioda vykreslování bodů a počet bod v jednom grafu.



Příloha 3.3 – Správce plovoucích grafů

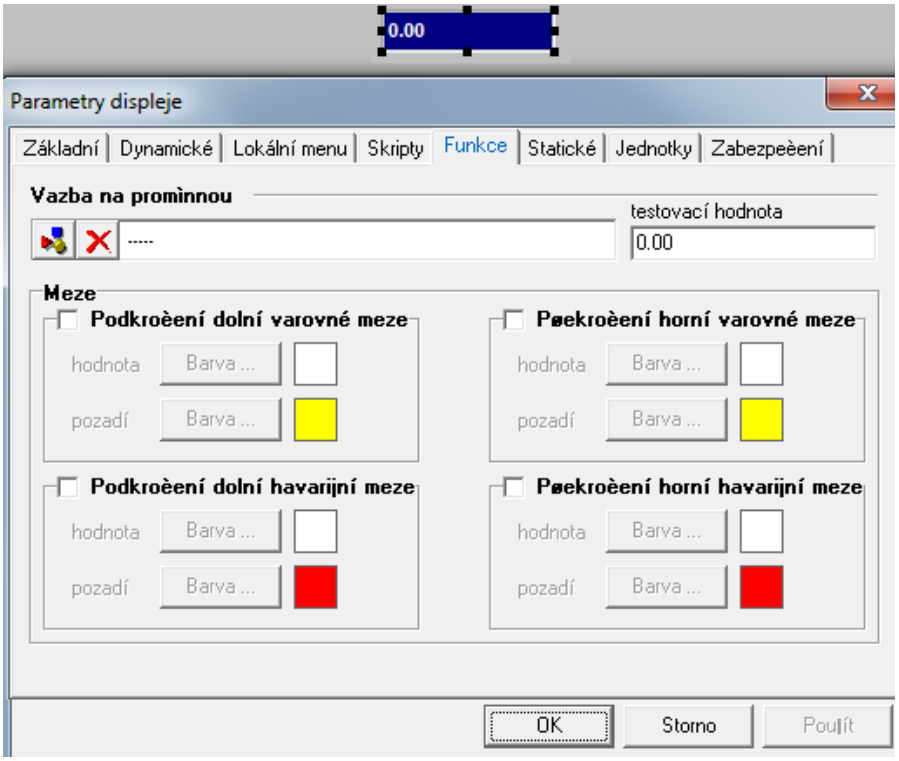
Pro export dat z vizualizace do textového formátu je ještě nutné nastavit *Správce tabulkových sestav*. Po otevření se opět pravým tlačítkem vybere složka *Tabulkové sestavy* → *Nová sestava*, kde po vytvoření se pravým tlačítkem vybere vytvořená sestava → *Přidat položky sestavy*, kde se vyberou proměnné, které se později použijí pro export. V pravé části se ještě přiřadí databáze, pro kterou bude daná tabulková sestava vytvořená.



Příloha 3.4 – Správce tabulkových sestav

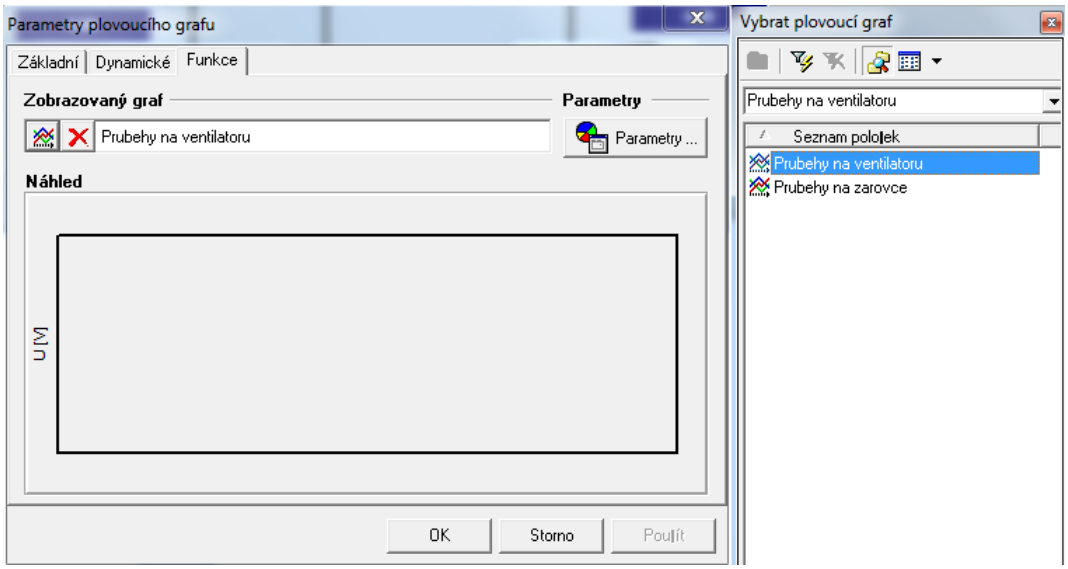
Všechny vytvořené správce je ještě nutné přiřadit do *Správce struktury projektu*, protože jinak by se s nimi nemohlo pracovat. Po otevření *Správce struktury projektu* je vše intuitivní, pouze stačí pravým tlačítkem vybrat příslušného správce, který byl vytvořen a vybrat položku *Propojit stanice*, kde se zobrazí všechny vytvořené položky pro daného správce a vybere se ta správná.

Po provedeném nastavení všech správců je možné vytvářet vlastní vizualizaci, uvedu zde, jak přidat obrázek, protože přidání obrázku není tak jednoduché, případně přiřazení proměnných prvkům vizualizace nebo přiřazení grafů. První prvek je displej, který je snadno k nalezení v horní liště, a po vybrání pravým tlačítkem daného displeje se vybere položka *Parametry displeje*, kde na záložce *Funkce* je možné přiřadit proměnnou, která se bude zobrazovat na displeji. Samozřejmě displej umožňuje mnohem více nastavení, jako typ zobrazení, barvu atd., ale nejdůležitější je přiřazení proměnné, aby vůbec něco zobrazoval.



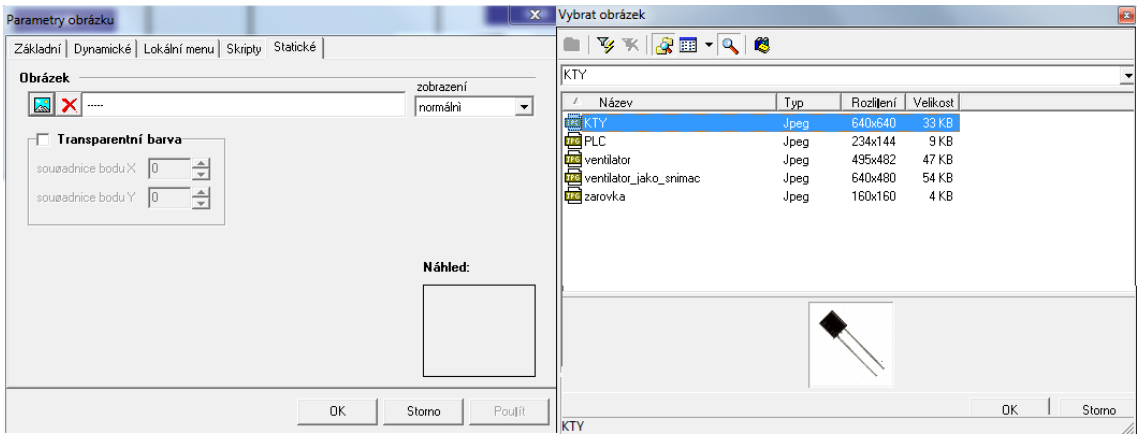
Příloha 3.5 – Vizualizační prvek – displej

Přidání prvku plovoucího grafu je jednoduché, opět se vybere z horní lišty příslušná položka a umístí se na hlavní plochu. Pravým výběrem plovoucího grafu, a následném výběru *Parametry plovoucího grafu*, lze na záložce *Funkce* přiřadit plovoucí graf, který se dříve vytvořil ve *Správci plovoucích grafů*.



Příloha 3.6 – Prvek plovoucí graf

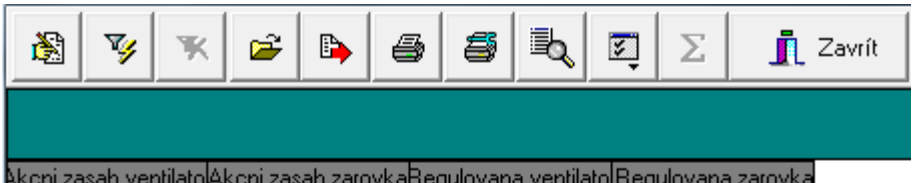
Pro přidání obrázku je také správce obrázků, ale není definovaný globálně, je potřeba tohoto správce přiřadit v aktuálním vizualizačním okně. Obrázek se přidá výběrem příslušné položky z horní lišty do hlavního okna, kde se obrázek následně vybere pravým tlačítkem a otevře se *Správce obrázků okna*. Tento správce slouží pro import obrázků do vizualizačního programu, proto je nutné nalézt obrázky na pevném disku PC a přiřadit je do správce a následně ikonkou Importovat obrázky je přiřadit do projektu. Po importu lze už snadno obrázky zobrazit, pravým tlačítkem se vybere prvek obrázku → *Parametry objektu*, kde pod záložkou *Statické* se vybere příslušný importovaný obrázek.



Příloha 3.7 – Přidání obrázku

Pro přenos dat z OPC klienta do textového formátu se v aplikaci Reliance Runtime vybere položka v horní liště *Připojené tabulkové sestavy*, kde po otevření podokna se vybere příslušná Sestava. Otevře se nové okno, kde jsou v řádcích v tabulce

zobrazené hodnoty sledovaných proměnných (příloha 3.8). Výběrem položky *Exportovat sestavu* a následném nastavení, jak se budou data zapisovat, se vybere umístění textového souboru a po potvrzení dojde k exportu dat do daného textového souboru (příloha 3.9).



Akcni zasah ventilato	Akcni zasah zarovka	Regulovana ventilato	Regulovana zarovka
0,44	2,95	0,95	2,78
0,39	2,89	1,03	2,77
0,35	2,84	1,1	2,75
0,4	2,95	1,01	2,76
0,45	3	0,93	2,78
0,44	2,96	0,95	2,79
0,4	2,91	1,01	2,78
0,38	2,89	1,05	2,77
0,41	2,88	0,99	2,81
0,45	2,86	0,94	2,84
0,42	2,81	0,99	2,84
0	0	0	0
0,4	2,68	1,02	2,88
0,4	2,68	1,02	2,88
0,82	3,13	0,28	3,06
0,33	1,97	1,16	3,1
0,77	2,98	0,39	3,07
0,68	2,8	0,57	3,06

Příloha 3.8 – Příklad tabulkové sestavy

```
Akcni zasah ventilator;Akcni zasah zarovka;Regulovana ventilator;Regulovana zarovka
0;0;0;0
0,39;0,83;0,97;3,39
0,38;0,81;0,99;3,4
0,35;0,74;1,05;3,39
0,36;0,77;1,02;3,39
0,38;0,8;0,99;3,43
0,4;0,85;0,96;3,45
0,38;0,81;0,99;3,44
0,35;0,73;1,05;3,39
0,34;0,72;1,06;3,43
```

Příloha 3.9 – Možný formát exportovaných dat